SAMS: Disk®

VOS3 Edition

Release 9.1
November 1998

Installation Guide

Sterling Software, Inc.
Storage Management Division
625 East Carnegie Drive, Suite 100
San Bernardino, California 92408-3510

SMDS300P001-91

Customer Support

For Users in the United States, contact:

Sterling Software, Inc.
Storage Management Division
10811 International Drive
Rancho Cordova, CA 95670
(800) 889-0226

For Users Outside the United States

For technical support please contact your local Sterling Software customer support representative.

Table of Contents

List of Figures

List of Tables

Chapter 1. Preinstallation

SAMS:Disk executes on Hitachi (or compatible) processors running under VOS3/ES, VOS3/AS, or VOS3/FS operating systems. Other operating systems are formally supported only if they are fully compatible with one of these.

This guide covers installation, customization, evaluation and maintenance of the Sterling Data Storage Management System, SAMS:Disk. An overview that outlines the steps is presented below. Each step contains detailed instructions.

You will probably follow most of the steps, but some of them will be skipped depending on your particular data center.

SAMS:Disk System Installation Summary

Installation Planning

☐ Planning Item 1: Disk Space and Naming Conventions

☐ Planning Item 2: APF-Authorized Libraries

☐ Planning Item 3: User SVC

☐ Planning Item 4: Auto-Restore CSA

☐ Planning Item 5: Link-list Library

☐ Planning Item 6: LPA Library

☐ Planning Item 7: Help Library

☐ Planning Item 8: Parameter Library

☐ Planning Item 9: Tailoring and Other Considerations

Installation

☐ Preparing the Environment

- Step 1. Distribution Tape Structure
- Step 2. Preparing for Multiple Releases of SAMS:Disk
- Step 3. Download the Related Installation Materials Library
- Step 4. Allocate SAMS:Disk Data Sets
- Step 5. Load SAMS:Disk

☐ Activating the SAMS:Disk System

- Step 1. Identify Files, Parmlib, and Activation Codes
- Step 2. Modify Symbolic Parameters in JCL Procedures
- Step 3. Customize JCL Procedures
- Step 4. Create a Files Definition Member in Parmlib
- Step 5. Initialize the Files Data Set
- Step 6. Convert pre 8.0D Files Data Set
- Step 7. Verify SAMS:Disk Enqueue Usage

☐ Activating SAMS:Disk Features

- Activating SAMS:Disk Security Features and Interfaces
  - Step 1. Select Password-Indicated Data Set Support Options
  - Step 2. Activate SAMS:Disk System Security
  - Step 3. Install SAMS:Disk Security Interface
- Activating VSAM Support
- Activating TSS Support
- Activating DASD Billing

Customization

- ☐ The SAMS:Disk SVC
  - SAMS:Disk SVC Modes
  - Tailoring the SAMS:Disk SVC
  - Installing the SAMS:Disk SVC
- ☐ The Auto-Restore Function
  - The Auto-Restore Method
  - Setting up Auto-Restore DASD Pools
    - VSAM Data Sets
    - Non-VSAM Data Sets
  - Customizing the TSS Auto-Restore Environment
  - Auto-Restore Implementation Guidelines
  - Auto-Restore Via VSAM Alternate Indexes and Path Names
  - Auto-Restore Restrictions
  - Additional Auto-Restore Considerations
- ☐ Customizing the TSS Support
  - Tape Units Concurrently Allocated
  - Tape Unit Name
  - Limiting TSS User Access
  - TSS Screening Exits for Archive and Restore
- ☐ Evaluation
  - Testing the SAMS:Disk System
  - Testing the Auto-Restore Feature

Installation Planning

The installation process will proceed much more smoothly with a little advance preparation. You should consider the following items before starting the actual installation.

Item 1: Disk Space and Naming Conventions

Make available enough disk space to load and install the system, and decide on a naming convention for the SAMS:Disk data sets. Disk space estimates for all libraries are documented in the program directory file on the installation tape.

The installation tape contains two categories of libraries:

System Libraries	—	LINKLIB library LPA library HELP library PROCLIB library
Product libraries	—	SAMS:Disk related libraries

Item 2: APF-Authorization

Load Module Libraries

Establish two APF authorized libraries in member JAAAPFnn of SYS1.PARMLIB for SAMS:Disk.

- Your SAMS:Disk Load Library
- Your SAMS:Disk Linklib Library

TSS Command Names

Establish APF authorization for the following command by placing its name in member JETCMDnn of SYS1.PARMLIB:

- RESTORE

Doing this in advance permits you to test executions immediately after downloading the distribution tape. Otherwise, you will have to wait for an IPL to authorize the libraries.

Item 3: User SVC

Add an entry for the SAMS Disk SVC to your system tables, consult your Hitachi SE for details on how to do this. It should be a type 3 or type 4 SVC, enabled for interrupts. Designating it as either restricted or non restricted is optional. Detailed information for installing the SVC can be found on page 33.

Item 4: Auto-Restore CSA

Ensure that approximately 41K bytes of ECSA (Extended Common Service Area) is available in your operating system if you are planning to install the SAMS:Disk auto-restore function. Detailed information for installing the SAMS:Disk auto-restore catalog management hook can be found on page 34.

Item 5: Link-list Library

Add the LINKLIB library to the LNKLSTxx member of your system parmlib. If you plan to copy the LINKLIB library into an existing library in the link-list, make sure that approximately 45k bytes of disk storage are available in that library.

Item 6: LPA Library

Copy the contents of the LPALIB library into an your existing LPA library, make sure that approximately 8k bytes of disk storage are available in that library. The LPALIB library contains the SAMS:Disk SVC, this must be copied to the system LPA dataset before IPL.

Item 7: Help Library

If you are installing the TSS interface, either concatenate the HELP library to the //SYSHELP DD in your TSS logon JCL, or if you plan to copy the HELP library into your system TSS help library, make sure that approximately 6k bytes of disk storage are available for six members that must reside in that library. These members are required for the TSS interface.

Item 8: Parameter Library

Save a copy of your PARMLIB so that you can copy your user-defined members after the SAMS:Disk installation process is complete. The following JCL, located in member PARMUPD1 in the installation library, can be used to accomplish this:

```
//JOBNAME  JOB (ACCT INFO)
//*
//*********************************************************************
//* JCL TO COPY PARMLIB MEMBERS YOU CREATED FOR TAILORING PURPOSES    *
//* FROM -- YOUR CURRENT (OLD) PARMLIB,                               *
//* TO ---- YOUR NEW PARMLIB FOR THE NEW RELEASE.                     *
//* WARNING: SAMS:DISK SYSTEM DEFINED MEMBERS IN THE NEW PARMLIB      *
//*          MUST NOT BE OVERLAID BY MEMBERS FROM THE OLD PARMLIB!    *
//*          COPY ONLY USER DEFINED MEMBERS, NOT SAMS:DISK            *
//*          SYSTEM DEFINED!                                          *
//*********************************************************************
//COPY    EXEC PGM=JSDPCPY,REGION=1024K
//SYSPRINT  DD SYSOUT=*
//SYSUT3    DD UNIT=SYSDA,SPACE=(TRK,3)
//SYSUT4    DD UNIT=SYSDA,SPACE=(TRK,3)
//SYSUT1    DD DISP=SHR,DSN=SAMS.OLD.PARMLIB   <-- OLD RELEASE PARMLIB
//SYSUT2    DD DISP=SHR,DSN=SAMS.DISK.PARMLIB  <-- NEW RELEASE PARMLIB
//SYSIN     DD DISP=SHR,DSN=SAMS.DISK.INSTALL(CPYNOREP)
```

Figure 1-1. Sample PARMUPD1 Member

Item 9: Tailoring and Other Considerations

Review the options and suggestions in the Tailoring Options section (beginning on page 49) in the *Systems Guide* because there are some additional options and considerations that you will probably want to review. Some important topics include:

- Archive and Backup Considerations on page 55.
- Processing PDSs with Anomalies on page 69.
- User-Specified Completion Codes on page 81.

Additionally, review the sections entitled *“Special Considerations”* (page 12) and *“General Restrictions”* (page 19), both in the *User’s Guide*, and implement any special processing controls indicated for your installation’s environment.

Chapter 2. Installation

This section of the SAMS:Disk Installation Guide provides step-by-step instructions for installing the basic SAMS:Disk system and activating various functions. Installation uses the programs JSDPCPY and LABUPDTE to download the distribution tape.

Preparing the Environment

Step 1. Distribution Tape Structure

SAMS:Disk is distributed on a standard label tape. The VOLSER of the distribution tape can be found on the external label of the installation tape.

Step 2. Preparing for Multiple Releases of SAMS:Disk

You may want to install this release of SAMS:Disk and continue to use an older release for your production environment. If you plan to run a previous release, you should consider the following:

- Be sure you do not install the new release modules into the same libraries as the old release modules. Doing so will cause a compatibility problem with one or both releases. To avoid this problem, ensure the DD statements in your download JCL point to the new release libraries.

- Throughly review JCL member DSK03REL of the installation library.

Step 3. Download the Stage Control Library

The JCL needed to install SAMS:Disk is provided in the related materials library on the distribution tape. Run the following JCL (member DSK03STG) to allocate the library and download it from tape.

```
//JOBNAME  JOB (ACCT INFO)
//*
//*********************************************************************
//* NAME: DSK03STG                                                    *
//*                                                                   *
//* SAMS:DISK INSTALLATION/MAINTENANCE JCL FOR HITACHI’S VOS3 SYSTEM  *
//*                                                                   *
//* INSTRUCTIONS:                                                     *
//*    CHANGE ’SAMS.DISK910’ TO INSTALL HLQ FOR SAMS:DISK.            *
//*    CHANGE ’VOLSER’ TO TARGET VOLUME.                              *
//*    CHANGE ’UNIT=CART’ TO A DEVICE SUPPORTING CARTRIDGE-MT.        *
//*                                                                   *
//*                                                                   *
//* NOTE:                                                             *
//*    THE //SYSUT2-FILE MUST NOT BE BLOCKED LARGER THAN 3200-BYTES   *
//*    BECAUSE SOME OF ITS MEMBERS ARE INPUT TO THE LINKAGE-EDITOR.   *
//*                                                                   *
//*********************************************************************
//*
//* STAGE CONTROL LIBRARY
//*
//STAGE   EXEC PGM=JSDPCPY,REGION=3072K
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DISP=OLD,DSN=DISK.RIMLIB,
//             VOL=SER=REL910,
//             UNIT=CART,LABEL=(2,SL)
//SYSUT2    DD DISP=(,CATLG),DSN=SAMS.DISK910.RIMLIB,
//             UNIT=SYSALLDA,VOL=SER=VOLSER,
//             SPACE=(3200,(300,15,100)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSIN     DD *
 COPY O=SYSUT2,I=((SYSUT1,R))
/*
```

Figure 2-1. JCL to Download the RIM Library

Step 4. Allocate SAMS:Disk Target Libraries

Customize and submit member DSK03ALC to allocate all of the SAMS:Disk target libraries.

Step 5. Generate the System

Customize and submit member DSK03GEN to load the target libraries from tape, link-edit the load modules into the target load libraries, and build the AutoLink database.

Activating the SAMS:Disk System

Below you will find instructions for activating the basic SAMS:Disk system. All steps are required; none are optional. The instructions include:

1. The SAMS:Disk Parameter Library (Parmlib) — Identify Files and Parmlib Data Sets to SAMS:Disk Functions

2. The SAMS:Disk JCL Procedures — Modify Symbolic Parameters in JCL Procedures and Customize JCL Procedures

3. The SAMS:Disk Files Data Set — Create a Files Definition Member in Parmlib and Initialize the Files Data Set

The SAMS:Disk Parameter Library (Parmlib)

SAMS:Disk uses control parameters to provide many different kinds of information needed for processing. Some indicate how SAMS:Disk is to operate, some define the format of reports, some define data sets for which SAMS:Disk is to skip processing, and others provide user-dependent information to SAMS:Disk functions.

These parameter lists are specified as members of a partitioned data set known as the parameter library, or parmlib for short. The parmlib implementation allows SAMS:Disk to better tailor its operation to your requirements. The contents of a parmlib member can easily be changed to reflect the desired options. Once they are changed, the next execution of SAMS:Disk will use them.

The parameter library is ready for use just as it was loaded from tape. The supplied members are regarded as parameter lists to be used for internal SAMS:Disk system functions and must not be changed (they are replaced with each new release, and occasionally with maintenance). Parameter information needed by your installation and users may be placed in new members of the parameter library that you create.

The PARMLIB section, beginning on page 437 of the *Systems Guide*, explains common SAMS:Disk parameter lists and the purpose of each. The sections for the other SAMS:Disk functions will further define the parmlib members that each uses. However, for the remainder of the installation process, only two members of parmlib are of primary concern. These are members SYSPARMS and FILEDEFN.

Member FILEDEFN is used to define the internal attributes of the SAMS:Disk files data set (“catalog” or “index” data set). This is discussed in a later installation step.

Member SYSPARMS is used to indicate your choice of processing options to SAMS:Disk. The Sysparms section, beginning on page 95 of the *Systems Guide*, describes all of the system parameters (sysparms) that are available to modify the way SAMS:Disk executes. All have system defaults; do not try to review every one of them now. The remaining installation steps and the customizing instructions will reference specific sysparms which should familiarize you with their use. As an example, the default retention period for data sets being archived is 30 days. To change this to 120 days, create a member called SYSPARMS in the parmlib data set and add one line that contains ARCRETPD0120 and/or RETRETPD0120.

Since the SAMS:Disk parameter library is a partitioned data set, it can be updated like any other PDS. If you do not have an online editor (i.e., such as ASPEN or TSS-edit), then use the Hitachi utility JSDUPDT for your updates.

Step 1. Identify Files, Parmlib, and Activation Codes

Several functions of SAMS:Disk need to know the names of the files data set and parmlib data set to be used within the function. The names of these data sets are made available to SAMS:Disk through the SET command.

In addition, indicators can be set to allow the use of preallocated files and parmlib data sets. Alternate files and parmlib data sets can then be used simply by allocating them prior to executing the desired function.

To provide the correct data set names and permit alternate data sets to be preallocated, use the following utility. A copy of this JCL can be found in member name DMSSET in the SAMS:Disk installation library.

```
//MODIFY  EXEC PGM=ADSMI002,PARM=ADSDM338
//STEPLIB  DD  DISP=SHR,DSN=SAMS.DISK.LOAD
//SYSLIB   DD  DISP=SHR,DSN=SAMS.DISK.LOAD
//ABNLDUMP DD  DUMMY
//CMDPRINT DD  SYSOUT=A
//MSGPRINT DD  SYSOUT=A
//PARMLIB  DD  DISP=SHR,DSN=SAMS.DISK.PARMLIB
//SYSPRINT DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
//SYSIN    DD  *
   SET .....
   Command Input
```

Figure 2-2. Utility to Identify Parmlib and Files

The SET command specifies the data set names and options to be used during some SAMS:Disk functions. An example format for the command and a description of its parameters follow. Only one command is accepted per execution:

`SET FILESDSN=,PARMSDSN=,ALTFILES,ALTPARMS,PARMLIBD,INIT=,SVCNBR`

FILEDSN=

The name of the SAMS:Disk files data set to use in the SAMS:Disk environment. This is a required parameter.

PARMSDSN=

The name of the SAMS:Disk parmlib data set to use in the SAMS:Disk environment. This is a required parameter.

ALTFILES

Specify this simple parameter to allow users to allocate DDNAME=FILES to their TSS session prior to executing SAMS:Disk command processors. This allows users to use other than the default files data set.

ALTPARMS

Specify this simple parameter to allow users to allocate DDNAME=PARMLIB to their TSS session prior to executing SAMS:Disk command processors. This allows users to use other than the default parmlib data set.

PARMLIBD

If your online environment already uses a ddname of PARMLIB (for something other than SAMS:Disk), specify this parameter to cause SAMS:Disk to use the alternate ddname of PARMLIBD.

INIT=

Specify this parameter to activate SAMS:Disk as indicated in your Installation Instructions letter.

SVCNBR=

This parameter may be used to indicate the SAMS:Disk SVC number, but it is no longer required for internal SAMS:Disk processing. This value should be a 3-digit SVC number in decimal form. The default value is 244.

The SAMS:Disk JCL Procedures

SAMS:Disk provides a set of PROCLIB JCL members for all functions in order to save time in the initial JCL preparation, reduce execution failures due to improper JCL, and greatly enhance the installation, testing and continued use of the SAMS:Disk functions. Some of these procedures may need to be modified to increase the size of the temporary and sortwork files, depending on the size of your FILES data set. The procedure library contains the following:

Table 2-1. Proclib JCL Members

Procedure Name	SAMS:Disk-Related Function
ARCHIVE	explicit archive
BILLING	non-VSAM DASD billing (select, accumulate)
COMPRES	PDS compression
DEFVOLS	list tapes needed for deferred restores
DERASE	delete deferred archive/restore requests
DIM	Dynamic Installation Manager
DMS	DSCL-invoked functions
DMSAR	auto-restore

Procedure Name	SAMS:Disk-Related Function
DMSGTF	diagnostic procedure used for problem resolution
DMSPOOL	tapepool update utility
DMSUTIL	miscellaneous diagnostics
EXTEND	DASD billing (extend totals and clear)
FMS	DSCL Recover
FRECOVER	Files data set forward recovery
IXCATLG	catalog archived DSNs for auto-restore
IXMAINT	index maintenance utilities
IXUPDATE	index update utilities
LISTD	list archive index
LISTREQ	list deferred archive/restore requests
MERGE	merge unexpired archive data sets
MERGCOPY	copy new archives created by merge
MIGRATE	sequential migration to tape
PDS2SEQ	merge PDS members into a sequential file
REBUILD	archive index rebuild
RECOVER	implicit recovery
RELOAD	files data set recovery
REORG	files data set reorganization
REPORT	special purpose non-VSAM reports
RESTART	restart after REORG shutdown
RESTORE	explicit restore
SMFRPT	data set report based on SMF data
THRSHMGR	volume threshold manager
TSTAR	Test a new/updated auto-restore function
UNLOAD	files data set backup
VSAMBILL	VSAM DASD billing (select, accumulate)
XCOPY	Disaster recovery extract utility

All of these procedures make consistent use of the following symbolic parameters and their recommended defaults.

Table 2-2. Symbolic Parameter Names and Defaults

SYMBOLIC	DEFAULT	EXPLANATION
P =	SAMS.DISK.INSTALL	library for control statements
Q =	SAMS.DISK	qualifier for SAMS:Disk data sets
S =	*	SYSOUT class default
W =	SYSDA	WORKFILE default unit name (must have a storage volume) U=

The procedures generate the following data set names if the recommended P and Q symbolic parameters are used:

Table 2-3. Data Set Names Generated by Procedures

SYMBOLIC	DEFAULT
&Q..PARMLIB	SAMS.DISK.PARMLIB (contains the system parameters and installation options)
&Q..LOAD	SAMS.DISK.LOAD (contains the executable load modules)
&Q..FILES	SAMS.DISK.FILES (contains the index of archived or backed up data sets)
&Q..ARCHPRIM	SAMS.DISK.ARCHPRIM (data set name used for the primary archive/backup tapes)
&Q..ARCHCOPY	SAMS.DISK.ARCHCOPY (data set name used for the duplicate copy archive/backup tapes)
&Q..MERGPRIM	SAMS.DISK.MERGPRIM (data set name used for the primary tapes created by merge)
&Q..MERGCOPY	SAMS.DISK.MERGCOPY (data set names used for the duplicate copy tapes from merge)
&U.&Q..WORKFILE	SAMS.DISK.WORKFILE (data set names used for the PDS compression workfile)
&P	SAMS.DISK.INSTALL (data set names used for the library containing needed control statements)

The procedures also use the following control statements that by default are retrieved directly from the SAMS:Disk installation library. They may be copied to any source image library, but the symbolic parameter P must then be modified to name the appropriate library.

Table 2-4. Procedures and Associated DD Statements

DD STATEMENT	PROCEDURE(S) THAT USE THEM
DMSORT1	ARCHIVE, DMS, DMSLRAC, MERGE, MIGRATE, RECOVER, REPORT, RETAIN
DMSORT2	MERGE
DMSORT3	EXTEND (DASD Billing)
DMSORT4	COMPRES
DMSORT5	DMS
DMSORT6	sort for non-DSNINDEX subfile records
DMSORT7	sort for DSNINDEX subfile records only
DMSORT8	DMS
DMSPASS1	REPORT
DMUNLOAD	MERGE
GTFPOPT	DMSGTF
RELOAD	REORG, RESTART and to reload subfiles
SMFSORT	SMFRPT

Step 2. Modify Symbolic Parameters in JCL Procedures

Each JCL procedure may be updated manually using ASPEN edit or any other editor you prefer. Alternatively, two utilities are provided in the Installation library that allows you to globally edit the JCL procedures:

1. **PROCUNLD -** Use this JCL utility against your procedure library to unload the procedures into a sequential file in JSDUPDT format. Use ASPEN edit or another editor to globally modify and change the procedures. For example: To change the first and second level qualifiers of your SAMS:Disk target libraries, you could use the following ASPEN edit command:

   ```
   1:$AA/SAMS.DISK/SYS2.SAMS.DISK/
   ```

2. **PROCRELD -** Use this JCL utility against your procedure library to reload the tailored procedures into your target procedure library.

Step 3. Customize JCL Procedures

The PDS Compress function uses a workfile to perform the compression. It is allocated by the //COMPWORK dd statement in the distributed COMPRES procedure, and must be big enough to hold the largest PDS selected for compression. The supplied primary and secondary space allocation quantities are adequate for processing most PDSs, even when limiting the workfile to a single volume. However, to compress a very large PDS, the space available for the workfile on one volume may not be sufficient. If you view this as a potential problem, update the COMPRES proc directly and change the value of the WRKVOLS= parameter, which defaults to 1. A larger value permits the workfile to extend to multiple volumes as needed. Set it to the maximum number of volumes you want to make available to the workfile. Your installation's configuration will limit the value you can assign to this parameter.

The distributed procedures for ARCHIVE, DMS, RECOVER, RESTORE and RETAIN all include dd statements for tape drives that have been commented out. Although these dd statements may be "uncommented" to cause system allocation routines to allocate the proper device when the job is initiated, SAMS:Disk default processing will dynamically allocate them when and as needed. Dynamic allocation is recommended because the actual need for the device depends upon the SAMS:Disk parameters that are specified for the job. Simulated functions probably won't need the drives, but when submitted in live mode, they will. Similarly, the SAMS:Disk ARCHIVE and RESTORE facilities have options to queue the requests rather than executing them immediately. Queuing the requests does not require the drives, but later processing of the queues will.

"TAPE" is used for the tape unit name and is not included as a symbolic parameter. You may need to update it to the appropriate value for your installation.

The region size in most of the procedures has been set at 5120K. Monitoring the actual memory use during the evaluation period may permit further reductions, based on the specific processing at your installation.

The SAMS:Disk Files Data Set

SAMS:Disk uses a single direct access data set, called the files data set, to record information related to the various functions you will be executing. The information stored in the files data set is divided up into sections called subfiles. The following table lists each subfile and describes its function.

Table 2-5. Files Data Set Subfiles

Subfile	Description
DSNINDEX	an index of all data sets in the archives
ARCHVOLS	an index of tape or disk volumes that contain archived data sets
DMSPOOLS	volser names of tapes (assigned to pools) that may be used during archive or backup runs
ARCHCMDS	a queue for user requests to archive or backup specific data sets
RESTCMDS	a queue for user requests to restore specific data sets
RETEXCLD	rearchive grace periods for restored data sets
DASDSPCB	DASD space billing records
MIGRECAT	recatalog information for sequential migration backup tapes
DMSPARMS	TSS dynamic (immediate) restore information
RACFENCD	RACF profile name cross-reference

The next two steps are required for all new users of SAMS:Disk. If you already have a files data set from a prior release, you may continue to use that files data set; however, you may wish to create and initialize a new files data set for testing purposes before you install this release into your production environment.

Step 4. Create a Files Definition Member in Parmlib

To create the files data set, first define its subfile characteristics to SAMS:Disk. A sample set of file definition entries is supplied in the parameter library. The only variable in each entry is the number of records (capacity) you expect to need in each subfile. The minimum is one. For your initial use, the supplied capacities should be more than sufficient, and will also provide the best performance. Copy the sample definitions (member FDSAMPLE in parmlib) into a new member of parmlib named FILEDEFN. This new member will be used to initialize your files data set.

Step 5. Initialize the Files Data Set

After your file definition member FILEDEFN has been created as outlined above, run the following JCL to allocate and initialize the files data set. Your output will consist of a status report based upon your definitions. You can find a description of the status report beginning on page 7 of the *Systems Guide*.

The following JCL can also be found as member FILEINIT in the installation library.

```
//FILEINIT EXEC PGM=ADSMI002,PARM=ADSDM100,REGION=2048K
//STEPLIB  DD  DISP=SHR,DSN=SAMS.DISK.LOAD
//ABNLDUMP DD  DUMMY
//CMDPRINT DD  SYSOUT=A
//FILES    DD  DSN=SAMS.DISK.FILES,
//             DISP=(,CATLG,DELETE),
//             UNIT=SYSDA,
//             DCB=(DSORG=DA),
//             SPACE=(CYL,10,,CONTIG)
//MSGPRINT DD  SYSOUT=A
//PARMLIB  DD  DISP=SHR,DSN=SAMS.DISK.PARMLIB
//SYSPRINT DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
```

Figure 2-3. Installation Library Member FILEINIT

Step 6. Convert pre 8.0D Files Data Set

If your current release of SAMS:Disk is 8.0D or below, you must convert your files data set into a different format. Please turn to page 77 for details.

Step 7. Verify SAMS:Disk Enqueue Usage

SAMS:Disk issues several types of enqueues during processing. Some enqueues are the result of normal system services such as dynamic allocation. Others are issued for internal SAMS:Disk functions. Turn to page 477 of the *Systems Guide* for more information about the enqueues, dequeues, and reserves issued by SAMS:Disk. If you use SAMS:Disk in a multi-system environment, make sure that all enqueues are propagated across all systems. Failure to do so may result in problems later.

Note: When a reserve against a SAMS:Disk files data set is converted to an enqueue, sysparm RSUPPRES needs to be set to a Y to function properly.

Activating SAMS:Disk Features

Below you will find instructions for activating the following SAMS:Disk features:

- SAMS:Disk System Security Features and Interfaces
- VSAM Support
- TSS Support
- DASD Billing
- Stand-Alone Restore

Activating SAMS:Disk Security Features and Interfaces

Step 1. Select Password-Indicated Data Set Support Options

In Hitachi's standard password support, when a password-indicated non-VSAM data set is opened, the operating system will prompt the TSS user (for online applications) or the master console operator (for batch jobs) for the password to the data set in question. Authority to continue will be based on the response to that prompt. For non-VSAM data sets, the password indicator bit is the DS1IND10 bit set (bit x'10' at offset 93 x'5D') in the format-1 DSCB.

SAMS:Disk has special code to avoid prompting the operator for the password on OPEN and SCRATCH of non-VSAM user data sets. This special code is used by SAMS:Disk only when sysparm PASSWORD is specified with a value of Y, when the PASSWORD parameter is included on the command (if it has an optional PASSWORD parameter), and when SAMS:Disk is running as an APF-authorized task. Most TSS sessions run non-APF-authorized, so they would not be exempted by SAMS:Disk from authority checking.

There are two substeps for installing the password-indicated data set support options. The substeps are:

1. Review the setting of sysparm PASSWORD. If you want SAMS:Disk to process password-indicated data sets, specify the sysparm value as Y. If you do not want SAMS:Disk to process password-indicated data sets, you may let it default to N.

2. Review the setting of sysparm PASSNEWN. If you want SAMS:Disk to allow the renaming of password-indicated data sets, specify the sysparm value as Y. If you do not want SAMS:Disk to allow the renaming of password-indicated data sets, you may let it default to N.

Step 2. Activate SAMS:Disk System Security

Security administrators should review SAMS:Disk sysparm TSOUSRID (page 176 of the Systems Guide), which applies to the TSS function.

Activating Miscellaneous Security Features

Certain SAMS:Disk parmlib members contain information that you may not want your users to override. The SYSPARMS member is one such example. This section describes how to restrict SAMS:Disk to a list of authorized parmlib data sets. You may activate this protection now or at a later date.

Parmlib Security — PARMAUTH

If you wish to restrict your users to a list of authorized SAMS:Disk parmlibs, activate the Parmlib Security Feature by installing user exit *“USERMOD5”* as follows:

1. Locate the source for the Parmlib Security Feature in PARMAUTH, located in the SAMS:Disk SAMPLIB library. Below is sample of that source:

```
PARMAUTH TITLE 'SAMS:Disk System parameter Data set security'
*******************************************************************
*         COMPILE ASEM=RENT,LKED=RENT                             *
*                                                                 *
* DESCRIPTION:                                                    *
*    This is a sample usermod used to tailor SYSPARM data set     *
*    specification security.                                      *
*                                                                 *
*    Read the PARMAUTH-macro prolog for specifications of options *
*    you may override.                                            *
*                                                                 *
 ******************************************************************
PARMAUTH PARMAUTH SECURITY=NO,                                      X
               SECURLIB=SYS1.PARMLIB,                               X
               SECURTBL=ZDMSPARM
         END
```

Figure 2-4. Sample source for PARMAUTH

2. In order to ensure that the changes you make to PARMAUTH are protected during future SAMS:Disk installs or maintenance, copy this member into the SAMS:Disk ASM library. The installation process, run later, expects to find the PARMAUTH source in the ASM library.

3. Customize PARMAUTH as follows:

   - Activate the Parmlib Security feature by specifying “YES” to the SECURITY= parameter. The default is *“NO”*, which deactivates the feature.

- Alter the SECURLIB= parameter as required. The value you specify is the name of the data set that will store the list of authorized parmlibs. The default is *“SYS1.PARMLIB”*. However, this may be any highly protected library available to SAMS:Disk with an LRECL of 80.

- Alter the SECURTBL= parameter as required. The value you specify is the name of the member that stores the actual list of authorized parmlibs. The default is *“ZDMSPARMS”*.

4. Create a member to store your authorized parmlibs. The data set must match that specified for *“SECURLIB=”*, and the member must match that specified for *“SECURTBL=”*. If you allowed SECURLIB= to remain at its default value, you must create the member *“ZDMSPARM”* in your cataloged SYS1.PARMLIB data set.

   To create a list of authorized parmlibs, use the sample SAMPZDMS provided for you in PARMLIB.

   To prevent a security exposure, only cataloged SAMS:Disk parmlibs may be authorized. If a user creates an unauthorized parmlib (or a data set with the same name on another pack), then tries to use the parmlib via the JCL VOL=SER= parameter, SAMS:Disk will note that the parmlib is uncataloged or incorrectly cataloged, issue a descriptive message and abend.

   Note: If you are installing a new release and have created a test parmlib, you must include this test parmlib in the list. If you wish to keep your production users from using your test parmlib, instruct your security package to give access to the test parmlib only to yourself.

USERMOD5

USERMOD5 must be installed using the JCL located in member DSK03USR in the SAMS:Disk RIMLIB library. Install USERMOD5 in the following manner:

1. Locate the install JCL in member DSK03USR in the SAMS:Disk RIMLIB.

2. Copy this member, DSK03USR into the SAMS:Disk ASM library.

3. In the ASM library, modify the JCL in the following manner:

   - Add your jobcard.

   - Change all occurrences of the name “SAMS.DISK” to your installation’s data set name qualifiers for the SAMS:Disk libraries.

- Change all occurrences of the name “SAMS.AUTO” to your installation’s data set name qualifiers for the SAMS:Disk LINKLIB and LPALIB libraries.
- Change all occurrences of “TEMPNAME” to “PARMAUTH”.
- Run the JCL. Check the condition codes in the SYSOUT to ensure a good run. If there were problems, look in the “SYSPRINT” or the “REPORT” DD output for the reasons - correct the problems and rerun the JCL.

4. After a successful run, SAMS:Disk attempts to read your SECURLIB= data set. If you allowed SECURLIB= to remain at its default value and your security package protects SYS1.PARMLIB, you must accomplish one additional step, and that is to make the list of authorized parmlibs available to SAMS:Disk. To accomplish this, do the following:

   - Grant READ access to each of your SAMS:Disk users via all programs from your SAMS:Disk load library.

     Note: If you have users using SAMS:Disk TSS functions under a TSS session that is not APF-authorized, these functions must obtain a shared enqueue to read SYS1.PARMLIB. If the shared enqueue can be obtained quickly, this should not be a problem as the read is very brief. However, if a shared enqueue cannot be obtained, the job goes into a wait state until the enqueue is available. SAMS:Disk tasks running APF-authorized do not use an enqueue on SYS1.PARMLIB, and are not affected.

Other Usermods

All other usermods can be installed using the installation instructions for USERMOD5, which begin on page 19. The usermods are located in the SAMS:Disk SAMPLIB library. The following table is a list of usermods applicable to the VOS3 environment:

Table 2-6. List of Usermods

SAMPLIB Member Name	Brief Description
TSOEXMPT	TSS exclusion/exemption table
DSNEXMPT	DSN exclusion/exemption table
JOBEXMPT	JOB exclusion/exemption table
PGMEXMPT	PGM exclusion/exemption table

SAMPLIB Member Name	Brief Description
TASKLIB	Used to modify the SAMS:Disk DDNAME for programs running in the TSS environment. Read the TASKLIB-macro prolog for specifications of options.
LINEMAX	Used to specify the page line-count, limit for the SAMS:Disk message processor. Read the LINEMAX-macro prolog for specifications of options you may override.
SYSOUTEX	Used to specify an exit routine for the SAMS:Disk message processor. Read the SYSOUTEX-macro prolog for specifications of options you may override.

DSK03USR

The following JCL can be used as a sample for all usermods in a Hitachi VOS3 installation:

```
//JOBCARD..............................................................
//* ****************************************************************************
//* *              HITACHI VOS3 USERMODS INSTALLATION                          *
//* ****************************************************************************
//* *  THIS JOBSTREAM, WHICH INCLUDES AN IN STREAM PROCEDURE, IS USED TO        *
//* *  ASSEMBLE AND LINKEDIT USERMODS INTO THE SAMS:DISK SYSTEM                *
//* *                                                                          *
//* *  PERFORM THE FOLLOWING STEPS TO INSTALL A USERMOD:                       *
//* *                                                                          *
//* *     1. COPY YOUR USERMOD SOURCE INTO THE SAMS:DISK ASM                   *
//* *        LIBRARY.                                                          *
//* *                                                                          *
//* *     2. COPY THIS MEMBER, DSK03USR, INTO THE SAMS:DISK ASM                *
//* *        LIBRARY AND MODIFY THIS MEMBER THERE.                             *
//* *                                                                          *
//* *     3. ADD YOUR JOBCARD.                                                 *
//* *                                                                          *
//* *     4. CHANGE ALL OCCURRENCES OF THE NAME   SAMS.DISK                    *
//* *        TO YOUR INSTALLATION'S DATA SET NAME QUALIFIERS                   *
//* *        FOR THE SAMS:DISK LIBRARIES.                                      *
//* *                                                                          *
//* *     5. CHANGE ALL OCCURRENCES OF THE NAME   SAMS.AUTO                    *
//* *        TO YOUR INSTALLATION'S DATA SET NAME QUALIFIERS                   *
//* *        FOR THE SAMS:DISK LINKLIB AND LPALIB LIBRARIES.                   *
//* *                                                                          *
//* *     6. CHANGE ALL OCCURRENCES OF THE MEMBER NAME  TEMPNAME               *
//* *        TO THE USERMOD NAME YOU ARE INSTALLING.                           *
//* *                                                                          *
//* *     7. RUN THIS JOB.                                                     *
//* *                                                                          *
//* ****************************************************************************
//USRMOD  PROC Q='SAMS.DISK',
//             M='TEMPNAME',
//             S='*'
//ASM      EXEC PGM=JOAXASM,REGION=1024K,
```

Figure 2-5. Sample source for DSK03USR

```
//             PARM='OBJECT,LIST,NODECK'
//SYSLIB   DD  DISP=SHR,DSN=&Q..MACLIB
//         DD  DISP=SHR,DSN=SYS1.MACLIB
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=SYSALLDA,SPACE=(4096,(500,100))
//SYSPRINT DD  SYSOUT=&S
//SYSLIN   DD  DSN=&&SYSLIN,UNIT=SYSALLDA,SPACE=(3120,(100,15)),
//             DISP=(NEW,PASS),DCB=BLKSIZE=3120
//SYSIN    DD  DISP=SHR,DSN=&Q..ASM(&M)
//*
//LKED    EXEC PGM=LNKEDT,
//             PARM='EX=EA,LD=ANY,NCAL,XREF,LIST,LET',
//             COND=((4,LT,ASM)),REGION=1024K
//SYSLIN   DD  DSN=&&SYSLIN,DISP=(OLD,DELETE)
//SYSLMOD  DD  DISP=SHR,DSN=&Q..PGMLIB(&M)
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=SYSALLDA,SPACE=(1024,(50,20))
//SYSPRINT DD  SYSOUT=&S
//    PEND
//* ************************************************************
//USRMOD   EXEC USRMOD,M='TEMPNAME'
//* ************************************************************
//AUTOLINK EXEC PGM=AUTOBIND,REGION=3072K,TIME=20,
//             PARM='MAP,LIST'
//* ************************************************************
//* *      AUTO-LINK STEP                                      *
//* *         TAKES MODULE FROM THE  SYSMLIN  DD STATEMENT     *
//* *         LOCATED BELOW AND CORRECTLY LINKS IT INTO THE    *
//* *         SAMS:DISK LOAD LIBRARY.                          *
//* ************************************************************
//* AUTO-LINK MEMBER(S)
//*
//STEPLIB   DD DISP=SHR,DSN=SAMS.DISK.LOAD
//REPORT    DD SYSOUT=*
//SYSPRINT  DD UNIT=SYSALLDA,
//             SPACE=(4840,(900,180)),DCB=BLKSIZE=4840
//LINKLIST  DD DISP=SHR,DSN=SAMS.DISK.LINKLIST
//SYSTERM   DD UNIT=SYSALLDA,SPACE=(121,(300,60))
//TERM      DD DISP=SHR,DSN=SAMS.DISK.TERM
//SYSLMOD   DD DISP=SHR,DSN=SAMS.DISK.LOAD
//ALTLNK    DD DISP=SHR,DSN=SAMS.AUTO.LINKLIB
//ALTLPA    DD DISP=SHR,DSN=SAMS.AUTO.LPALIB
//SYSLIB    DD DISP=SHR,DSN=SAMS.DISK.PGMLIB
//LNKT      DD DISP=SHR,DSN=SAMS.DISK.LNKT
//SYSUT1    DD UNIT=SYSALLDA,SPACE=(CYL,3)
//SYSLIN    DD UNIT=SYSALLDA,SPACE=(3200,(300,60))
//AUTODATA  DD DISP=SHR,DSN=SAMS.DISK.AUTOLINK.IMOD
//AUTOLIB1  DD DISP=SHR,DSN=SAMS.DISK.AUTOLINK.AIX.IOBJ
//SYSUDUMP  DD SYSOUT=*
//SYSIN     DD DISP=SHR,DSN=SAMS.DISK.RIMLIB(DSK03ALT)
//SYSMLIN   DD *
TEMPNAME
/*
```

Activating VSAM Support

Follow these steps to activate the VSAM support.

1. Specify sysparm VSAMSUPP with a value of Y to activate the VSAM support.

2. A SAMS:Disk master password is available to allow processing of any password-protected cluster. For security reasons, it is supplied and documented in a separate enclosure. Review that documentation for use in your installation.

3. Review sysparm VSONLINE if your installation uses DMF/EF Type-2 catalogs and you have VSAM data sets defined on offline DASD volumes. For related information, please review the sysparm description for *VSDSPACE* on page 186 of the *Systems Guide*.

4. JSCVSUT uses the JSCUT1 and JSCUT2 dd statements in the RESTORE, RECOVER and DMSAR procedures to dynamically allocate work space when building a VSAM alternate index. You can use one of two approaches to control where this space is allocated:

   a. Modify the data set names to direct the allocation to a desired VSAM or DMF/EF Type-2 catalog, and modify the VOL=SER to point to volume(s) where work space can be allocated. You will also have to rename dd statements IDCUT1 and IDCUT2 to JSCUT1 and JSCUT2 respectively.

   b. Supply the necessary sysparms to cause SAMS:Disk to dynamically allocate the needed dd statements prior to invoking BLDINDEX processing.

   If you choose the first method (supplying the JCL statements directly), review sysparms VSDEFCAT, VSBIXPSW and VSBIXCAT for consideration. The main drawback to this approach is that it forces all restore and recover jobs that use the same data set names for the work files to be single-threaded (that is, each job must wait until the previous restore has completed and released the exclusive enqueues on the work file data set names). This can cause major processing delays and frustration if there are many restore jobs waiting to be serviced. Also, the enqueues are left outstanding even if no VSAM data sets need to be restored.

   If you choose the second method (to leave the JSCUTx dd statements out of the JCL), simultaneous restore jobs can take place (as long as they don't require the same archive tape). When SAMS:Disk restores a VSAM alternate index and determines that BLDINDEX processing needs to be invoked, it will check to see if the JSCUTx dd statements are allocated. If they are not, it will generate a unique name for each of the dd statements and allocate them accordingly. Two sysparms govern the allocation of these dd statements.

VSIDCUTPxxxxxxxx—is used to specify the high-level node SAMS:Disk is to use for the work file name. For instance, if VSIDCUTPLABS is specified, the generated work file name will begin with LABS. If a value is not specified, the high-level node of the first alternate index being built will be used. This sysparm would be most useful in installations that have not converted to DMF/EF Type-2 and therefore will need to have the work data set placed on specific volume(s) owned by the VSAM catalog pointed to by the high- level node of the data set name.

VSIDCUTVxxxxxx...yyyyyy—is used to specify the volume(s) to be used for the work data set. Up to ten volumes can be specified in the list. If more than one volume is desired, make sure that each of the volumes is specified as exactly six characters (for example, 'VSIDCUTVWRK800WRK802WRK806'). The names that SAMS:Disk generates have the form:

```
PREFIX.jobname.Dyyddd.Thhmmss
```

Table 2-7. VSIDCUTP Generated DSNAME

Qualifier	Meaning
PREFIX	the high-level node of the alternate index being built or the VSIDCUTP parameter
jobname	the job name being used for the current restore or recover operation
Dyyddd	the current date in Julian format
thhmmss	the current time in hours, minutes, and seconds

5. If your installation has vendor products--or internally developed application programs--that generate non-standard VSAM entry-sequenced data sets (ESDS) or linear data sets (SVSAM), review sysparms VSACCESS and VSARCFMT. These data sets must be archived in a control interval image copy format rather than in logical record format.

Activating TSS Support

Follow these instructions to activate the support.

TSS command processors are available to perform the functions as listed below.

Table 2-8. TSS Command Processor Available Functions

Module	Function
DARCHIVE	Queue a request to archive a data set
DRESTORE	Queue a request to restore a data set
DERASE	Erase a queued archive or restore request
LISTDMS	List the SAMS:Disk index of archived data sets
LISTREQ	List the status of queued archive/restore requests
RESTORE	Restore a data set immediately (uses dynamic tape allocation). This command processor must also be designated as privileged.

Step 1. Make SAMS:Disk Available to TSS

This process has been partially completed, since you have already installed the TSS Command Processors, and control program ADSMI202 into one of the system linklist libraries.

To complete the process, modify your TSS logon procedure to allocate a ddname of DMSOSLIB that points to the SAMS:Disk load library; that is:

```
        //DMSOSLIB DD DISP=SHR,DSN=SAMS.DISK.LOAD
```

This step is required to give the TSS Command Processors access to other SAMS:Disk programs.

The SAMS:Disk control module ADSMI202 always looks for the DMSOSLIB library first, and will find all of the needed SAMS:Disk modules there without searching any other libraries.

Step 2. TSS Help Text

The Help text members were placed into the SAMS:Disk TSS Help library during the installation. Ensure that this library is allocated to your TSS session.

Step 3. The TSS Dynamic Restore Command Processor

If you want to permit the use of the dynamic RESTORE command, follow the instructions below to activate this feature. For more information on the dynamic RESTORE command, turn to page 414 in *User's Guide*.

The TSS dynamic RESTORE command processor uses a subfile called DMSPARMS, defined within the files data set. This subfile was described in the

FDSAMPLE member in parmlib, and was created when you defined and initialized the files data set.

The dynamic restore command processor must be designated as privileged or authorized in the VOS3 environment, due to its use of the ENQ and ALLOCATE SVCs. Update your SYS1.PARMLIB(JETCMDxx) member to include “RESTORE”. After the changed member has been activated by an IPL, any TSS user may do dynamic restores. TSS mount authority is required if the data set being restored is archived to tape.

Activating DASD Billing

Follow these instructions to activate the support.

1. A RESTART data set is required for DASD billing. It is used to clear the billing file and in proper recovery after an abend. It is referenced by the //RESTART dd statement in the EXTEND procedure, and defaults to DSN=&Q..RESTART.

2. Create the RESTART data set, and allocate it with the following attributes:

Table 2-9. Restart Data Set Attributes

Attribute	Value
DSORG	PS
RECFM	F
LRECL	96
BLKSIZE	96
SPACE	(CYL,(2,2))

Chapter 3. Customization

The SAMS:Disk product has been developed so as to provide the DASD administrator with a great deal of flexibility. In the pages that follow, several of the more commonly used customization options will be discussed.

Each customization step discussed will include the proper JCL and instructions to install.

As part of your planning for installing one or more customizing options, you should verify that the JCL provided conforms to your installation's JCL procedures. The Assembler JCL provided uses step names of "C" and "L". Your Assembler procedure may use "ASM" and "LKED".

The following customizing options will be discussed in this section:

- The SAMS:Disk open SVC and applicable open zap.
- The auto-restore function.
- Defining TSS customization options.

The SAMS:Disk SVC

The purpose of the SAMS:Disk SVC is to update a data set's VTOC entry when the data set is being opened, recording the last used date, for example. The SAMS:Disk SVC has the following advantage over the standard operating system VTOC update code: exemption entries are provided such that the SAMS:Disk management tasks themselves don't cause data sets to appear used.

Zaps to one or more open modules must be applied to call the SVC to do the updating. Contact the HITAC service center and request the special-modification for SAMS:Disk SVC. The HITAC service center address is

```
5030, Totsuka-cho Totsuka-ku,
Yokohama-city, Kanagawa, Japan 224
Tel:045-862-8892
Fax:045-865-9063
```

The SAMS:Disk SVC maintains the last change bit and last used date fields. It will also maintain some extra SAMS:Disk-defined fields, if desired, and provides the facilities to overcome the deficiencies as described below.

The two fields maintained by the operating system appear to provide the information that is needed to help manage disk storage, and SAMS:Disk functions can examine these fields and execute without the SAMS:Disk SVC being installed.

However, from a practical point of view, the SVC will probably be required for your implementation for the following reasons:

a. The SAMS:Disk storage management tasks themselves open user data sets to perform the management functions; for example, backup, migration, compression, etc. The operating system by itself will mark these data sets as being used whenever these management functions are run, and therefore prevent other functions, such as archive, from finding and processing data sets that are truly inactive. That is, a potentially large number of data sets appear used and are kept on disk merely due to storage management jobs routinely being run. The SAMS:Disk SVC exempts the management jobs from causing these updates.

b. The SAMS:Disk SVC can also record the date whenever the change bit is turned on (mod date). This may be useful in report information and, as explained in the BACKUP/ARCHIVE section of the *User's Guide*, it also provides a convenient and low-overhead means of truly eliminating redundant backup copies of data sets (that full pack dumps or other more simplistic techniques often create).

c. The SAMS:Disk data set utilization (DSU) report loses much of its usefulness since it reports on the additional fields maintained by the SAMS:Disk SVC.

SVC Requirements - If the SAMS:Disk SVC is to be installed, a user SVC of type 3 or 4 must be provided by the installation. It should be enabled for interrupts and may be designated as either restricted or non-restricted. (See *"Planning Item 3"* on page 4 for more details.)

SVC Integrity and Security - Before any fields are modified in the DSCB, the SVC verifies that the OPEN/CLOSE/EOV WORKAREA does contain a F1-DSCB. It also checks to insure that the caller is in key 0 and in supervisor state.

VTOC Fields Maintained by the SAMS:Disk SVC - If open modules are zapped to execute the SAMS:Disk SVC, the following fields are maintained in the F1-DSCB for each data set.

Table 3-1. VTOC Fields Maintained by SVC

Field #	DESCRIPTION	FORMAT	BYTES	OFFSET (DEC)
1	Last use date	ydd	3	75 SU 60
2	Last modify date	ydd	3	48
3	Last used job name	char	8	62
4	Count of updates to VTOC entry	bin	4	78

Field #	DESCRIPTION	FORMAT	BYTES	OFFSET (DEC)
5	Change bit	bit	x'02'	93 SU 60
6	SVC mode	char	1	103

Note: You must run SAMS:Disk 9.1 with the OPEN SVC from release 9.1. You cannot run with an OPEN SVC from a prior release.

SAMS:Disk SVC Modes

The SAMS:Disk SVC is distributed with a control switch called SVCMODE that affects which fields are maintained and how often.

SVCMODE=1

This mode of operation makes the minimum number of updates to the VTOC entry (F1-DSCB) for a data set, yet maintains all of the fields. The SVC source is distributed in this mode. It causes the F1-DSCB to be updated only if:

1. the last used date needs to be changed
2. the change bit needs to be set on (change bit turned on when data set opened for output), and MODDT will also be updated

Note that (1) will cause only one update per day, and (2) will cause only one per day unless the change bit is being turned off more often than that (for example, running incremental backup two or more times per day).

Field 4 shows the number of times the F1-DSCB has been updated, not necessarily the number of times a data set has been opened.

SVCMODE=2

This mode of operation causes updates under exactly the same conditions as mode 1, but only the standard fields (1 and 5) plus the mode field itself (6) are maintained. This is an operating system "look-alike" mode, with the benefit of the exemption tables.

SVCMODE=3

This mode of operation causes the F1-DSCB to be updated every time the data set is opened. Field 4 shows the number of times the data set has been opened.

Be aware that this technique causes many more updates of the VTOC entry, and is known to cause occasional problems with partitioned data sets, made evident by duplicate TTR directory entries and loss (overlay) of a member update. This usually occurs only for PDSs that are accessed very heavily and by two or more concurrent users. Shared DASD environments without a "cross-system enqueue package" are especially susceptible.

Tailoring the SAMS:Disk SVC

In order to use the SAMS:Disk SVC, you must tailor it for your installation. You do that by setting options in module ADSVOSOP, assembling it, and linking it with module ADSMVS60, the SAMS:Disk SVC. The steps to accomplish this are:

1. Locate member ADSVOSOP in the SAMS:Disk INSTALL library. This member contains the assembler source code for module ADSVOSOP. Module ADSVOSOP contains run-time options for the SAMS:Disk SVC.

2. Set your desired SAMS:Disk SVC options by modifying the options macro calls in member ADSVOSOP.

3. Tailor and submit the JCL in member ASMVOSOP in the SAMS:Disk INSTALL library. This job will assemble module ADSVOSOP and link it with ADSMVS60, the SAMS:Disk SVC.

```
TITLE 'ADSVOSOP - RUNTIME OPTIONS FOR ADSMVS60 - VOS3 VERSION'
         SPACE
*-----------------------------------------------------------------------
* GENERAL OPTIONS AND SETTINGS
*-----------------------------------------------------------------------
         SVCOPTS SYSTEM=HITACHI,       EXECUTION SYSTEM                 X
               DSCBCODE='HITACVOS3',   FORMAT 1 DSCB SYSTEM CODE        X
               SVCMODE=4               SAMS:DISK SVC MODE
         SPACE
*-----------------------------------------------------------------------
* PROGRAM NAME EXCLUSION TABLE
*-----------------------------------------------------------------------
         SVCPGM BEGIN                  BEGIN OF TABLE
         SVCPGM ENTRY,PGM=ADSMI        SAMS:DISK PGMS - DO NOT REMOVE!
         SVCPGM ENTRY,PGM=JSFDCDP      DCDP
         SVCPGM ENTRY,PGM=JBHSS00      DMFHSS
         SVCPGM ENTRY,PGM=JUXVSS       DMFVSS
         SVCPGM END                    END OF TABLE
         SPACE
*-----------------------------------------------------------------------
* DSNAME EXCLUSION TABLE
*-----------------------------------------------------------------------
         SVCDSN BEGIN                  BEGIN OF TABLE
         SVCDSN ENTRY,DSN=SYS8         1980'S SYSTEM TEMPORARY DATASETS
         SVCDSN ENTRY,DSN=SYS9         1990'S SYSTEM TEMPORARY DATASETS
         SVCDSN ENTRY,DSN=SYS0         2000'S SYSTEM TEMPORARY DATASETS
         SVCDSN ENTRY,DSN=**SYSUT      IEHMOVE TEMPORARY DATASETS
         SVCDSN END                    END OF TABLE
         SPACE
*-----------------------------------------------------------------------
* JOB NAME EXCLUSION TABLE
*-----------------------------------------------------------------------
         SVCJOB BEGIN                  BEGIN OF TABLE
******** SVCJOB ENTRY,JOB=MYJOB        SAMPLE ENTRY
         SVCJOB END                    END OF TABLE
         SPACE
*-----------------------------------------------------------------------
```

Figure 3-1. Sample INSTALL Member ADSVOSOP

```
* VOLSER EXCLUSION TABLE
*---------------------------------------------------------------------
         SVCVOL BEGIN                    BEGIN OF TABLE
******** SVCVOL ENTRY,VOL=MYVOL          SAMPLE ENTRY
         SVCVOL END                      END OF TABLE
         SPACE
*---------------------------------------------------------------------
* DIAGNOSTIC JOB NAME TABLE - INCLUSION TABLE
*---------------------------------------------------------------------
         SVCDIAG BEGIN                   BEGIN OF TABLE
         SVCDIAG ENTRY,JOB=DMSTEST       STANDARD ENTRY
         SVCDIAG END                     END OF TABLE
         SPACE
         END   ADSMVSOP
```

Installing the SAMS:Disk SVC and Hitachi OPEN Module ZAP

In order to use the SAMS:Disk SVC, you apply a ZAP to Hitachi VOS3 OPEN module. The ZAP will cause the Hitachi OPEN routine to call the SAMS:Disk SVC at the right time and place to modify the OPEN process. The steps to accomplish this are:

1. Work with the Hitachi Service Center to get the appropriate ZAP applied.

2. Tailor the SAMS:Disk SVC by following the instructions in ’Tailoring the SAMS:Disk SVC’.

3. Install the SAMS:Disk SVC on your system as SVC 244. To do this you must (1) Copy module ADSMVS60 (after tailoring) into SYS1.LPALIB, and (2) Update SYS1.PARMLIB to indicate the existence of the SAMS:Disk SVC as #244. If you want to use a different number for the SAMS:Disk SVC, you will need to coordinate this with the Hitachi Service Center so that the correct SVC number is called.

4. IPL your system. Be sure to do an MLPA or CLPA to initialize the link pack area.

The Auto-Restore Function

One of the more important reasons an installation purchases any data storage management system is to minimize the amount of DASD space allocated to data sets that are rarely or never accessed. This can be accomplished in SAMS:Disk by both explicit archival and implicit archival based on last used date. Typically, the more constrained an installation is for DASD space, the shorter the time allowed for a data set to be unused prior to archival. As the unused time window shrinks, more and more data sets will be archived that do get accessed infrequently. This is especially common for data sets that get referenced only on a monthly or yearly basis.

Although this archival scheme accomplishes one of the DASD manager's primary functions--ensuring enough DASD free space exists for day-to-day operations--it can also create a less than amicable relationship with the data center's users if not implemented properly. For instance, if a system is set up to archive all data sets not used within two weeks, those data sets that are only accessed once a month will be archived between each job cycle.

The user must either respond to this problem by restoring all required data sets prior to a job run, or artificially referencing the data sets periodically to avoid having them archived. In the first case, an extra workload is created for the user, and in the second case the DASD manager is prevented from doing an effective job. Although the DASD manager could exempt these data sets from being archived by SAMS:Disk, this may not be a good alternative since the data sets could then remain on DASD long after the system that uses the data sets is removed from production. And if an installation is critically short of free DASD space, it may not have the luxury of letting seldom- used data sets tie up valuable DASD space between job cycles.

So how does an automatic restore capability help to solve this potential conflict between the DASD manager and the end user? By allowing data sets to be archived by the DASD manager and then restored automatically by SAMS:Disk if they are required by an application at a later time. This solves the DASD manager's problem of keeping free space available, and also relieves the user of the burden of restoring required data sets prior to cyclical job runs.

The Auto–Restore Method

SAMS:Disk provides a Catalog Management hook to perform auto-restores. The previous S213-04 exit is not recommended and is no longer documented.

Catalog Management Hook:

SAMS:Disk dynamically installs a module that intercepts catalog management requests to inititate auto-restores. Since many catalog management requests do not require auto-restores, this module must decide dynamically which requests to intercept and which to pass on.

A catalog locate is usually issued by a routine trying to establish the existence of a data set. If the locate is successful, the data set exists; if it fails, it either doesn't exist or it's not cataloged. The key for the catalog management hook to go to work is when the locate is issued from a selected function (dynamic allocation, for example) and the return from catalog management is successful, but the data set is cataloged to the SAMS:Disk pseudo-volume. (The pseudo-volume--the SAMS:Disk default name is ARCIVE--is an imaginary volume to which SAMS:Disk optionally recatalogs a data set when it is archived and scratched, to help identify the data set as being archived.) When these conditions are met, SAMS:Disk will automatically restore the data set and then return to the requestor of the locate the real volume to which it was restored. In this way the locate requestor is never aware that a restore operation took place. We'll discuss how to catalog data sets to the pseudo-volume later in these customization instructions.

You will notice that in the preceding description of the catalog management hook we said that only specific functions will cause auto-restores to take place. For instance, it would not be desired to automatically restore all data sets referenced by an JSCVSUT LISTCAT job. Instead, these locates are ignored by the catalog management hook so that the output listing will show a volume of ARCIVE for the data set. However, SAMS:Disk will intercept locate requests from dynamic allocation, job initiation, and certain TSS requests.

If you attempt to delete a data set that has been cataloged to the SAMS:Disk pseudo-volume, the data set will be auto-restored.

Installing the Auto-Restore Function

The auto-restore function should be implemented only after a thorough review of the following installation procedure. Be sure to complete each step in the process before continuing to the next.

1. Review the procedure named "DMSAR" in the proclib library shipped with your SAMS:Disk system. In particular, review the following items:

   Note: If you rename the DMSAR catalog procedure, you must also change the value of sysparm ARTASKNM (the default name is DMSAR). Otherwise the support won't work!

   a. Review the dispatching priority assigned to the procedure and make any necessary adjustments based on your installation's requirements.

   b. Supply the proper data set name for the STEPLIB.

   c. If you desire dynamic tape allocation, leave the asterisk in column 3 of the ARCHIVES DD statement. If you do not want dynamic allocation to be performed by SAMS:Disk, remove the asterisk. The preferred method is to let SAMS:Disk dynamically allocate the tape device, since this will allow requests that do not require tape devices to be processed immediately, even if no tape devices are available. This condition would occur for those data sets that have been archived to disk, have no record in the SAMS:Disk archive index, or that are rejected by the user screening exit ARESPREX.

      If you put an ARCHIVES or ARCHIVER DD statement in your JCL, do not specify the DSN= parameter. If you specify a data set name, no concurrent auto-restores can take place. In addition, you may not be able to run auto-restores at the same time as SAMS:Disk backup, archival, restore or recover jobs.

If the archives reside on disk rather than tape, the appropriate disk archive data set will be dynamically allocated with no operator intervention required.

d. Review the IDCUT1 and IDCUT2 DD statements. These statements were discussed in *"Activating VSAM Support"* on page 23.

e. If your installation maintains multiple files data sets and you desire to make them all available to the auto-restore function, you should use the MFILES DD statement instead of the FILES DD statement. Add each files data set name to the MFILES concatenation, up to 256 files data sets are supported. The order of the concatenation determines the order of search for the data set. The first files data set that contains an index record for the archived data set will be used, even if another files data set contains a more recent version of the data set. Therefore, it is very important that files data sets be specified in the proper order.

   Note: The FILES DD statement will be ignored if there is also an MFILES DD statement in the procedure. Also note that concatenated files data sets are not supported with the FILES DD statement.

f. Ensure that the DMSAR proc is in a proclib accessible during an VOS3 START command; for example, SYS1.PROCLIB.

g. Users with security packages should note that since the SAMS:Disk auto-restore started task cannot be identified by a JOB statement or a LOGON procedure, most security packages have a Started Procedures Table to associate a started task name with a user ID and possible group name. If your security package protects resources accessed by started tasks and you would like to auto-restore data sets protected by your security package, you must create an entry in your Started Procedures Table to associate the SAMS:Disk auto-restore started task name DMSAR with a user ID and possible group name.

   For TRUST, the Started Procedures are defined by the STC parameter in the TRUST initialization parameters.

2. Review the documentation for sysparms ARESUNIT, ARESUNIC, and ARESUNIn (if multiple types of cartridge devices are used). If the defaults are unacceptable, you may specify their values in your production sysparms.

3. Review the documentation for sysparm ARTAPEOK. This sysparm controls the degree to which tape mounts will be accepted--if at all--from auto-restore tasks.

4. Review the documentation for user exit ARESPREX. This user screening exit can selectively reject auto-restore requests, and can also perform logging activities.

5. When you are installing the catalog management hook, you may want to use pool support as well. Pool support allows you to dynamically decide the volume to which to restore a data set, based on a combination of pool definitions and an optional user exit. Review the topic *"Setting Up DASD Pools for Auto-Restore"* on page 43. This does not have to done at installation time. You may want to install the basic auto-restore facilities first and add pool support later.

6. The catalog management hook of SAMS:Disk is installed with a started task. A system IPL is not required. You install the hooks by issuing the following command from the operator's console:

   ```
   S DMSAR,DMSAR=INSTALL
   ```

   Output is directed back to the operator's console. When the interface is successfully installed, this started task will end.

   Implementation Notes:

   a. You must issue this command on each CPU for which you want SAMS:Disk's auto-restore function implemented.

   b. The SAMS:Disk catalog management hook hook is removed each time you IPL. Therefore, you must use the above procedure to reinstall the hooks when you IPL. Consider installing the command in the automatic start-up procedures.

      You can check if the SAMS:Disk catalog management hook is installed by issuing the operator command:

      ```
      S DMSAR,DMSAR=STATUS
      ```

      or simply:

      ```
      S DMSAR
      ```

      You can remove the SAMS:Disk catalog management hook by issuing the operator command:

      ```
      S DMSAR,DMSAR=REMOVE
      ```

      If you do issue this removal command, there are several processing steps of which you should be aware:

After a remove command is accepted, the SVC table entry for SVC 26 is modified to point to its original addresses. This disables the hook. The removal task waits eight seconds so that current requests can complete, then DMSAR deletes its hook from CSA memory. The hook is completely removed and the started task terminates.

c. If you plan to use a SAMS:Disk pseudo-volume name other than ARCIVE, it must be specified through sysparm RECATVOL. Do not select a pseudo-volume name that is a real volume name, or likely to become one. If your pseudo-volume name is a real volume, you will not be able to reference your data sets on that volume through the catalog. Set the pseudo-volume name as the value for the sysparm RECATVOL.

d. An auto-restore diagnostic facility exists within the Catalog management hook which can display applicable data used to determine if an auto restore is to be performed. These diagnostic messages are available if you have installed the Test Auto-Restore Interface as documented below, and if your job name is equal to the test jobname (default is TESTDISK). If you would like to change this default jobname, simply zap the jobname you prefer. Refer to the TSTHOOKS member in the INSTALL library and installation instructions for the Test Auto-Restore Hook for further information.

   Similar diagnostic facilities are provided for the open SVC module. The default jobname for the other diagnostic facilities is DMSTEST, so you may desire to make this the same name for all functions. The ability to change the job names for the other functions will be discussed where applicable.

Test Auto-Restore Hook Interface

Overview

This function lets you install a *new* or *different* release of the Auto-Restore operating system hook simultaneously with your production Auto-Restore hook. The Test Auto-Restore Hook will only operate on jobs/tasks which have a certain predefined name or DD statement. It also allows you to reproduce problems and receive diagnostics through the use of sysparm ARDIAGNM. Your production Auto-Restore hook will continue to operate on all other jobs. This lets you safely test changes or upgrades to your Auto-Restore function while you leave your production SAMS:Disk system running.

How it Works

SAMS:Disk Auto-Restore Hook have program logic at their entry points which determines if each instance of the hook is to run or not. This determination can be based on either a specific job/task name or DD statement. In the case of the Test Auto-Restore Hook, care has been taken to ensure that the production hook will run only when the test hook doesn't run, and vice versa.

The Test Auto-Restore Hook will only operate on jobs/tasks which have a certain predefined name as their job/task name, or on jobs which include a certain pre-defined DD statement name. This is configurable via the following 2 system parameters:

- ARJOBEXC — the value of this system parameter is the pre-defined job name. It can also be a task name (i.e., a userid). For detailed information, review the sysparm description for *ARJOBEXC* on page 123 of the *Systems Guide*.

- ARDDNEXC — the value of this system parameter is the pre-defined DD statement. For detailed information, review the sysparm description for *ARDDNEXC* on page 122 of the *Systems Guide*.

Components of the Test Auto-Restore Hook Interface

The Test Auto-Restore Hook Interface consists of the following components:

Table 3-2. Test Auto-Restore Components

Component	Explanation
TSTHOOKS member in the INSTALL library. (Hook customization job.)	This job creates the Test Auto-Restore Hook. This job copies the production hook modules to test names and then ZAPs them so that they use the test job name and test Auto-Restore PROC. You need to run this job even if you just want to use the default options.
TSTAR member in the PROC library. (Test Auto-Restore PROC.)	This PROC is the same as the DMSAR PROC, except that it has been modified for use by Test Auto-Restore. This PROC is started by the Test Auto-Restore hook in order to perform a test Auto-Restore.
TSTEXMPT member in the INSTALL library. (Hook exempt job.)	This job refreshes the production Auto-Restore Hook to begin excluding the job/task name and/or the DD statement name based on the value specified for sysparms ARDDNEXC and ARJOBEXC.

Component	Explanation
TSTINSTL member in the INSTALL library. (Hook install job.)	This job dynamically installs the Test Auto-Restore Hook and begins to accept auto-restores from the job/task name and/or DD statement name based on the value specified for sysparms ARDDNEXC and ARJOBEXC. You do not need to IPL the system to install this hook.
TSTSTATS member in the INSTALL library. (Hook status display job.)	This job displays the status of the SAMS:Disk Auto-Restore operating system hook on the system console.
TSTREMOV member in the INSTALL library. (Hook remove job.)	This job dynamically removes the Test Auto-Restore Hook from your system. You do not need to IPL your system to remove this hook.
SVC 26 Hook Module.	Module TSTAR010 (a modified version of module ADSAR010).

Preparing to Use the Test Auto-Restore Hook Interface

There are several things which you must do before you use the Test Auto-Restore Hook feature:

1. Tailor the following jobs provided in the INSTALL or PROC libraries on your SAMS:Disk distribution tape: TSTHOOKS, TSTINSTL, TSTREMOV, TSTSTATS, TSTEXMPT, and TSTAR. You will need to tailor the job accounting, load library name (DD=STEPLIB), and the parameter library name (DD=PARMLIB).

2. Tailor the Q= parameter in the TSTAR PROC. This parameter provides a data set name prefix for the SAMS:Disk libraries in your system.

3. You must select the job name and/or the DD statement you want to have trigger the Test Auto-Restore feature. This is accomplished through the ARJOBEXC and ARDDNEXC system parameters. After selecting a job name and/or DD statement, supply these sysparms to

the TSTEXMPT member in the form on a sysparm override as shown in the following figure:

```
//JOBNAME  JOB (ACCT INFO)
//*****************************************************************
//** REFRESH THE PRODUCTION VERSION OF THE CATALOG MANAGMENT HOOK **
//** TO EXCLUDE THE "TSTAR010" TEST HOOK (SEE TSTINSTL).          **
//*****************************************************************
//DIMCAT   EXEC DIM,CMD=REFRESH,SVC=26,HOOK=ADSAR010
//MSGPRINT  DD  SYSOUT=*
//PARMLIB   DD  DISP=SHR,DSN=SAMS.DISK.PARMLIB
//SYSPARMS  DD  *
ARJOBEXCDMSTJOB
ARDDNEXCDMSTDDN
//*****************************************************************
//** REFRESH THE PRODUCTION VERSION OF THE HSM HOOK               **
//** TO EXCLUDE THE "TSTHS001" TEST HOOK (SEE TSTINSTL).          **
//*****************************************************************
//DIMHSM   EXEC DIM,CMD=REFRESH,TYPE=ESR,SVC=(3,24),HOOK=ADSHS001
//MSGPRINT  DD  SYSOUT=*
//PARMLIB   DD  DISP=SHR,DSN=SAMS.DISK.PARMLIB
//SYSPARMS  DD  *
ARJOBEXCDMSTJOB
ARDDNEXCDMSTDDN
/*
```

Figure 3-2. Sample JCL for the TSTEXMPT member

When this job is executed, it refreshes the production Auto-Restore Hook to begin excluding the job/task name and/or the DD statement name based on the value specified for sysparms ARDDNEXC and ARJOBEXC.

CAUTION: The same values you specify for sysparms ARJOBEXC and ARDDNEXC in the TSTEXMPT job, must also be specified in the TSTINSTL job. Otherwise, unpredictable results will occur.

Installing the Test Auto-Restore Hook Interface

The Test Auto-Restore Hook Interface can be installed by carefully following these step by step instructions:

1. Locate and submit the TSTHOOKS member you customized. This job creates the Test Auto-Restore environment by copying your production Auto-Restore modules to test names.

2. Locate and submit the TSTEXMPT member you customized. This job refreshes the production Auto-Restore Hook to begin excluding the job/task name and/or the DD statement name based on the value specified for sysparms ARDDNEXC and ARJOBEXC.

3. Locate and submit the TSTINSTL member you customized. This job dynamically installs the Test Auto-Restore Hook and begins to accept

auto-restores from the job/task name and/or DD statement name based on the value specified for sysparms ARDDNEXC and ARJOBEXC.

Operating the Test Auto-Restore Hook Interface

One of the objectives of this interface is to allow you to activate an Auto-Restore environment at a release level different from that of your production system. The Test Auto-Restore system you just installed will only operate on the job/task name and/or DD statement name that match the values you set for ARDJOBEXC and ARDDNEXC sysparms.

For example, with sysparm ARDDNEXC set to a value of TESTAR, the Test Auto-Restore Hook Interface will be invoked for all jobs displaying the TESTAR DD statement as illustrated in the following figure:

```
//JOBCARD  JOB ...
//*---------------------------------------------------------------------------
//*   THIS JOB WILL INVOKE THE SAMS:DISK TEST AUTORESTORE FACILITY
//*   (TSTAR), VIA DDNAME. 'TESTAR' VALUE SHOULD MATCH THE ARDDNEXC
//*   VALUE IN INSTALL LIBRARY MEMBER TSTINSTL.
//*---------------------------------------------------------------------------
//STEP1   EXEC PGM=JSCVSUT
//TESTAR   DD  DUMMY
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  PRINT IDS(ARCHIVE.DATASET)
/*
```

Figure 3-3. Sample JCL to invoke Test Auto-Restore

Note: The ARDDNEXC will only operate successfully on archived data sets that are referenced dynamically.

The convenient quality of this environment is that you can now thoroughly test this new Auto-Restore system without impacting your production system.

Sysparm ARDIAGNM

There is a diagnostic feature in this interface that may help you solve problems related to Auto-Restore. To activate this feature, simply assign a job name value to sysparm ARDIAGNM. Setting this sysparm instructs the Test Auto-Restore Interface to issue diagnostic messages to each job that enters the system with this job name.

The following figure illustrates some of the typical messages routed to SYSLOG when the ARDIAGNM sysparm is activated:

```
ADSAR010 DIAG01: SUPPL=00001152 7F6B4238 009DD9A8 009DD998
ADSAR010                  00000000 00000400 00000000 00000000
ADSAR010 DIAG02: PROGRAM INITIATING CALL WAS: IEFBR14
ADSAR010 DIAG03: ENTNAME=ISPDLH1.A.APARLIB
ADSAR010 DIAG04: CATNAME=SSL801ICFCAT.VSSL801
ADSAR003 DIAG99: RESTORE INITIATED BY INTERFACE
DMS2987 DATA SET CATALOGED TO SAMS:DISK PSEUDO-VOLUME ARCIVE
DMS2987 SAMS:DISK HAS ARCHIVED ISPDLH1.A.APARLIB
ADSAR003 DIAG99: AUTORESTORE BY START CMD=DMSAR
S DMSAR.DMS00052,,,ARCB=00BFC4AC,ASID=00052,RTIME=1532532  ** SAMS:DISK AUTO-RESTORE **
DMS2993 DATA SET SUCCESSFULLY RESTORED
```

Figure 3-4. Sample SYSOUT of ARDIAGNM

For detailed information, review the sysparm description for *ARDIAGNM* on page 122 of the *Systems Guide*.

Member TSTREMOV

Once your testing has completed, you need to shutdown the Test Auto-Restore Hook Interface. Doing so will reverse what was excluded from your production Auto-Restore while the Test Auto-Restore Hook were active.

The Test Auto-Restore Hook Interface can be removed by carefully following these step by step instructions:

1. Locate and submit the TSTREMOV member you customized. This job dynamically removes the Test Auto-Restore Hook.

2. Issue the REFRESH command to your production Auto-Restore system. This command sets sysparms ARJOBEXC and ARDDNEXC back to their default values, which deactivate them. The following command may be used as an example of the REFRESH:

   ```
   S DMSAR,DMSAR=REFRESH
   ```

Setting Up DASD Pools for Auto-Restore

This section applies only if you have installed—or plan to install—the catalog management hook. The use of the pool support is completely optional.

The purpose of the pool support in the Auto-Restore facility is to allow the maximum flexibility possible in determining the target volume to which the data set will be restored. Although this capability is not essential to the basic function of Auto-Restore, it does relieve the DASD administrator of a lot of the manual effort required to manage free space distribution among packs. Perhaps the best way to explain its advantages is to show what management would be like without it.

Without pool support, any data set that is Auto-Restored would be required to go back to the volume from which it came. Two problems become obvious:

1. What happens if the volume no longer exists?

2. What if there is not enough space for the data set to fit or the available space is to fragmented?

The answer is that the restore would have to fail and the task that requested the restore would fail. This is not too serious if a TSS user caused the restore, but it can cause a lot of grief if a production job abends because it couldn’t gain access to the data set.

Pool support within the Auto-Restore facility is accomplished separately for VSAM and non-VSAM data sets.

For VSAM Data Sets:

For VSAM data sets, pooling is technically complex, and must be accomplished by an allocation control product. It is not done within SAMS:Disk itself. If an allocation control product is not used, Auto-Restore must place VSAM data sets back on the volumes from which they were archived. If you do not have an allocation product available, you may consider using the RESPRIEX user exit, described on page 235 of the *Systems Guide*.

For Non-VSAM Data Sets:

For non-VSAM data sets, pooling may be accomplished through either or both of two methods. The first method is by using an allocation control product. The second method is accomplished within SAMS:Disk’s Auto-Restore facility, and can be used alone or in conjunction with an allocation control product.

Several advantages become apparent when volume pooling is implemented, such as:

- The data set can be restored to any volume in the pool that has sufficient free space. Each volume in the pool will be searched until one with sufficient free space is found, or until it is determined that none of the volumes in the pool has enough free space.

- If a volume (or a string of volumes) becomes unavailable, you could specify an entry in the volume pool that redirects all restores to that volume to a different pool of volumes.

- You can reserve a set of special volumes used only for Auto-Restores as its own pool. This would allow you the capability of managing this pool differently than perhaps your TSS or production packs.

- You can code your own user exit to decide the pool to which to restore, based upon criteria unique to your installation.

Pool support can be thought of as the algorithm used to determine the destination volume for a data set being Auto-Restored. It consists of several possible steps, depending on options you specify. Let's look at a schematic of the process, then we'll come back and fill in the details.

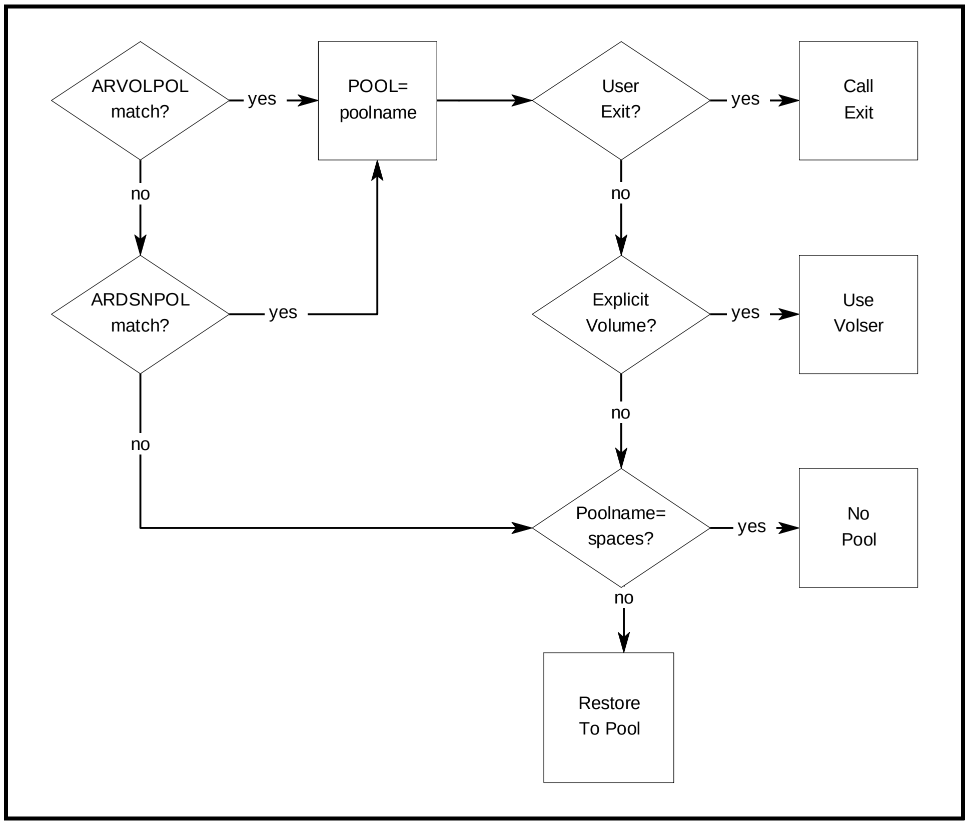

Figure 3-5. Example of DASD Pool Support for DMSAR

The following table illustrates a typical example of how the syntax works for DASD Pools. This table is then followed by a series of explanatory notes to help describe the illustration.

Table 3-3. Example of DASDPOOL Syntax

ARVOLPOL	ARDSNPOL	DASDPOOL
'POOLLABSLABS83'	'POOLTESTA.B.C'	'LABS80POOLLABS'
'POOLLABSLABS8/'	'POOLLABSD.E./'	'LABS81POOLLABS'
'POOLTESTTEST??'	'POOLLABSD./'	'LABS83POOLLABS'
'POOLWORKWORK/'	'POOLTEST???.*.TEST/'	'WORK01POOLWORK'
'POOLLABSLABS80'	'POOLWORK/'	'TEST01POOLTEST'

Note 1: Extensive rules for coding parmlib entries can be found on page 444 of the Systems Guide.

Note 2: All pools referred to in ARVOLPOL and ARDSNPOL must be defined in the member DASDPOOL in your parmlib data set. The DASDPOOL member is described in detail on page 444 of the *User's Guide*.

Note 3: All standard SAMS:Disk pattern matching capabilities are supported in both ARVOLPOL and ARDSNPOL members. The first entry that causes a match will cause its corresponding pool name to be used, even if there are other entries in the member that would match. This means that you should start each of these members with the most specific entries first and have the least specific entries entered last.

Note 4: Pool capability for VSAM data sets is supported by an allocation control product, not by the methods just described.

Note 5: If you wish to use an allocation control product for non-VSAM data set pooling, you should create an ARVOLPOL with at least one volume in it (a volume that would normally be empty or used for temporary data sets only). This is for two reasons:

1. SAMS:Disk will always check to see if the needed volume is online and has space available before attempting to allocate a data set it is restoring. If the original volume is offline or does not have enough space, SAMS:Disk will fail the restore without attempting to allocate the data set. No allocation control product would get control. By supplying an ARVOLPOL, you provide SAMS:Disk with an online volume to which it can attempt to make allocations.

2. An ARVOLPOL will give you a means of seeing how well your allocation control product rules are doing. By monitoring the volumes defined in this ARVOLPOL, you can determine which data sets are not being controlled by your rules. You may then update your rules to prevent similar problems from occurring later.

Going back to the schematic, we start at Step (a) to determine if a source volume pool match is found. There are actually two steps to this process, which also applies to Step (b). First SAMS:Disk will retrieve sysparm ARVOLPOL. If you have not specified this sysparm, a no-match condition is assumed. If you do specify the sysparm, the value you supply for it must be the name of the member in your parmlib that contains the pool entries. For instance, if a value of ARVOLPOLVOLPOOL is specified, you must code your entries for this pool in the member VOLPOOL in your parmlib. Assuming this pool is defined, SAMS:Disk will compare the data set's original source volume to each explicit or pattern volume in the table. If a match is found, the pool name associated with that entry will be used and processing will continue to Step (d). If no match is found, or if sysparm ARVOLPOL is not specified, processing goes on to Step (b).

Step (b) follows the same general logic as Step (a). You notify SAMS:Disk of the presence of a data set name pool by specifying sysparm ARDSNPOL with the member name that contains the pool associations. In this pool, SAMS:Disk will compare each entry to the data set name being restored. As soon as an explicit or pattern name is matched, that pool name will be used and processing will continue with Step (d). If no match is found, the pool name is initialized to spaces (Step (c)), indicating that no pool name will be used in the restore.

We are now at Step (e), which is an optional user exit that can be coded. If you want to write a special pool selection routine, here is where you would add it. SAMS:Disk will call the exit (Step (f)) when sysparm ARPOOLEX is specified with the module name to be called. For more information, see the User Exit description for ARPOOLEX on page 208 of the *Systems Guide*). With this exit you can see if any pool was selected, based on the entries in the ARDSNPOL and ARVOLPOL members. If no match was found or neither of the members were defined, the pool name is blanks (Step (i)); otherwise it will contain the name of the pool that will be used (Step (k)). You may change the pool name to another valid pool name or it can be blanked out to suppress pool support for the restore. You also have the option of passing back an explicit target volume (Step (g)), which will override any pool specification (Step (h)). The diagram shows that there is no requirement to use the pool support that SAMS:Disk provides--just by coding the user exit you can customize pool support.

If after the processes above you exit with the name of a pool, this pool must be defined in the member DASDPOOL in your parmlib. The eligible volumes to which the data set can be restored are all the entries in this member that have the same pool name.

At this point you may want to read the RESTORE/RECOVER section in the *User's Guide*, beginning on page 209. There you will find a detailed description of the rules SAMS:Disk follows in determining the volume to which a data set is restored.

Customizing the TSS Auto-Restore Environment

This section applies only if you have installed the catalog management hook. If you have installed the hook but have not attempted any restores from a TSS environment, you should do so at this time. It will be much easier to conceptualize the modifications about to be described if you are familiar with how the support works in its default mode.

In its default mode the TSS user is prompted for answers to the following questions:

1. DO YOU WANT TO RESTORE THE DATA SET? (yes/no) If the user replies anything other than “no”, the next question is asked.

2. DO YOU WANT AN IMMEDIATE OR A DEFERRED RESTORE? (I/D) If the user responds anything other than “deferred”, the next question is asked.

3. DO YOU WANT TO WAIT FOR THE RESTORE TO COMPLETE? (yes/no) If the user answers “no”, the terminal will be unlocked and the user is notified when the restore is complete. If the user responds anything other than “no,” the terminal remains locked until the restore is complete, unless a tape mount is required (and allowed by sysparm ARTAPEOK). If a tape mount is required, the user is prompted one more time with the following question:

4. DATA SET IS ARCHIVED TO TAPE. DO YOU STILL WANT TO WAIT? (no/yes) If the user responds “yes”, the terminal will remain locked until the restore is complete. If the user responds anything other than “yes”, the terminal will unlock and the user will be notified when the restore is complete. The Auto-Restore task will begin before the user responds to this prompt, and the response has no effect on the progress of the Auto-Restore job.

This series of questions can be very confusing to some users. If you want to suppress any or all of the questions, you can. This is done by installing user exit *"USERMOD8"* as follows:

1. Locate the source for ADSUMOD8 in the library associated with the DMSASM DDDEF. Below is a sample of that source:

```
ADSUMOD8 TITLE 'SAMS:Disk Auto-Restore options data'
*******************************************************************
*         COMPILE ASEM=RENT,LKED=RENT                             *
*                                                                 *
* DESCRIPTION:                                                    *
*  This is a sample usermod used to specify SAMS:Disk Auto-Restore*
*   options.                                                      *
*                                                                 *
 *   Read the ADSUMOD8-macro prolog for specifications of options *
*   you may override.                                             *
*                                                                 *
*******************************************************************
ADSUMOD8 ADSUMOD8 TIMEOUT=30,  Start timeout for DMSAR-STC          X
               READAUTH=YES,   Restore authorization check (YES/NO) X
               MSG2971=,       Do you want to restore? (Y/N)        X
               MSG2972=,       Immediate or Deferred? (I/D)         X
               MSG2973=,       Do you want to wait for tape? (Y/N)  X
               MSG2978=,       Archived-to-tape, wait? (Y/N)        X
               MSG3362=Y,      Uncatalog for TSS UNCAT/DEL? (Y/N)   X
               MSG3755=        Restore active, wait? (Y/N)
        END
```

Figure 3-6. Sample source for ADSUMOD8

2. In order to ensure that the changes you make to ADSUMOD8 are protected during future SAMS:Disk installs or maintenance, copy this member into the source library associated with the //USERASM dd statement in USERMOD8.

3. Customize ADSUMOD8 as follows:

   - The TIMEOUT= parameter defaults to 30 seconds before the STC passes control back to the USERID. After this wait period is exceeded, message 2977 is issued and the TSS user (or console operator in the case of a batch job) is asked if the Auto-Restore should be terminated. A “Y” response will terminate the Auto-Restore. Any other response will extend the wait period an additional 30 seconds. If your installation is prone to resource conflicts which slows task initiation, you may want to consider increasing this wait period and eliminate this additional prompt. This can be done by increasing the value of this parameter.

   - The READAUTH= parameter has no effect on VOS3.

   - The MSGnnnn= parameters are used to customize the default responses to the TSS prompts. Specify a "Y" to pre-answer the TSS response as YES. Specify a "N" to pre-answer the TSS response as NO. Specifying the "null parameter" will cause auto-restore to prompt the TSS user for a response. For MSG=2972, specify an "I" to pre-answer an immediate restore, a "D" to pre-answer a deferred restore, or the "null parameter" will cause auto-restore to prompt the TSS user for a response.

4. Save your updates. line.

USERMOD8

USERMOD8 must be installed using Assemble and Link. Sample JCL is provided for you in member USERMOD8 of the install library. Install USERMOD8 in the following manner:

1. Locate the source for USERMOD8 in the install library. Below is a sample of that source:

```
===> This JCL is still under development <===
```

Figure 3-7. Sample JCL for USERMOD8

2. Customize USERMOD8 by supplying the appropriate information for the following:

   - jobcard information
   - SAMS:Disk link library.
   - SAMS:Disk macro library.
   - The data set containing your modified USERMOD8 (ddname USERASM).

3. Submit USERMOD8.

4. Refresh the Auto-Restore hook on all systems for this change to take effect. Issue the following command to accomplish this:

```
S DMSAR,DMSAR=REFRESH
```

Auto-Restore Implementation Guidelines

The hardest task in implementing the Auto-Restore function is in determining how much time should be allowed to elapse between the time a data set is last referenced and when it should become a candidate for archival. If this time window is too large, few archivals will take place and a severe shortage of DASD space will probably occur. On the other hand, if the time window is too small, too many archivals may take place. If this happens, the cost of archiving and restoring the data sets will probably exceed any benefits realized by the temporary DASD space savings.

Obviously there are too many variables to determine an absolute rule in calculating the best time to archive a data set based on last use date. The best method appears to be by trial and error. First pick a period of time that seems reasonable based on current knowledge of the user environment. Then set up a monitoring period and keep totals on the number of auto- restore requests being processed on a per-shift basis. If this number stays sufficiently low during the trial evaluation, and no shortage of DASD space is evidenced during this period, then you have probably struck upon a good balance between archiving and restoring. Your evaluation period should run through your center's month-end processing cycle to verify that this load does not produce an unreasonable number of restore requests. If it does, your archival scheme probably needs to be re-evaluated.

One issue that must be considered on the archival side is the increased catalog usage due to the need to recatalog data set to the SAMS:Disk pseudo-volume so they can be automatically restored.

One facility that can help in the maintenance of your catalogs is the specification of sysparm UNCATPSU with a value of Y. This causes both MERGE and IXMAINT to issue an uncatalog (if the data set is cataloged and does not have a F1-DSCB on the source volume) when the last DSNINDEX record for the data set is deleted from the archive index. We highly recommended that this sysparm be specified with a Y if you are using the Auto-Restore function.

The established SAMS:Disk customer faces the problem that all data sets that were archived under an old system of “scratch, uncatalog” are not accessible by the Auto-Restore function. To assist these customers, a procedure has been implemented to catalog data sets that are found in the archive index but that are no longer cataloged and do not have a F1-DSCB on the source DASD volume. This procedure, named IXCATLG, is supplied with SAMS:Disk and is documented in Appendix B.

Auto-Restore via VSAM Alternate Indexes

When a VSAM cluster is archived and recataloged to the SAMS:Disk pseudo-volume, alias entries are also defined for each of the alternate index and path names, as well as for each of the components' names for the cluster. This enables the auto-restore function to be invoked when you reference the base cluster, any of its components, an alternate index, or path name associated with the base cluster.

Restore processing removes the catalog entry to the pseudo- volume and all of the aliases associated with it, and redefines the correct VSAM catalog entries. Should the restore process fail, SAMS:Disk attempts to re-create the initial status by recataloging the cluster to the pseudo-volume and associating the proper alias names to it.

Auto-Restore Restrictions

If the catalog management hook is installed, the following restriction applies:

- A reference to an uncataloged archived data set in a batch job with volume and unit information hard coded in the JCL will fail with a JCL error.

Additional Auto-Restore Considerations

In multiple CPU environments with cross-system enqueue packages, you must ensure that all enqueues are propagated across all CPUs. For auto-restores in particular, enqueues for DMSAUT0, DMSAUT1, DMSAUT2, DMSAUT3, and DMSAUT4 must be propagated accurately. These enqueues ensure that data integrity is maintained.

Precautions in Changing Immediate to Deferred Requests:

Sysparm ARTAPEOK (with a value of D) and user exit ARESPREX (by setting the return code to H’1’) both provide the ability to change an immediate restore request into a deferred request; that is, place the Auto-Restore request in the restore queue rather than process it immediately. The sysparm option will automatically defer a request only if it is for a TSS user and a tape mount is required. Batch jobs and restores from disk archives are not affected. The user exit is more general, however, and can defer any of the various request types.

There is also a slight exposure that you should consider before using either of them. As you will see from the description below, the exposure is extremely low under most circumstances, especially with the sysparm option, but indiscriminate use of the user exit option might cause a problem.

The concern exists only if two jobs and/or TSS users happen to want the exact same archived data set at the “same” time. SAMS:Disk Auto-Restore processing for the second request detects that another Auto-Restore task (serving the first request) is already processing the same data set. The second job is then placed in a wait state (if it’s batch), or informed that the restore is in progress, and to try again later (if it’s a TSS user).

If the sysparm option is used and the first request is from TSS and the second one from batch, the batch job will fail because of its assumption that the Auto-Restore for the TSS user was actually restoring the data set, when in fact it just placed it in the queue.

If the user exit is used to force a request to be queued instead of restoring it immediately, the results for the second job of two simultaneous requests can be as just described, without regard to the TSS or batch status of either requestor!

To summarize, the exposure exists only if two jobs/users attempt to access the same archived data set at the same time. This probability is very low. If the sysparm option is used, the probability is further reduced by the fact that the first job must be a TSS request, and the second a batch job.

Rejecting Auto-Restore Requests Via Exclusion Tables

The same user exit mentioned above, ARESPREX, can also be used to reject an Auto-Restore request. The exit has access to considerable data regarding the restore itself, and therefore provides a great deal of flexibility in the decisions that can be made, and consequently should be considered first if you have needs in this area.

Another method also exists, but it provides much more limited flexibility, and requires the assembly and linking of exclusion tables into a SAMS:Disk module. ADSAR003 is responsible for starting the DMSAR Started Task. This module is statically linked into ADSAR010, the catalog management hook. Before ADSAR003 even starts the restore task, it has the ability to screen the request and reject it, based upon either the dsname that is needed, the name of the executing program, or the jobname.

The programs and/or dsnames that are to be excluded are provided in several CSECTS statically linked into ADSAR010 and available to ADSAR003. These CSECTS define exclusion tables for programs, data sets, jobnames, and TSS USERIDs.

If you have program names, data set names, or jobnames that can always be excluded from Auto-Restore, then add them to these exclusion tables. This is accomplished by defining the desired exclusion table in a user assembly process and linking these into the Auto-Restore hook, ADSAR010 and ADSHS001.

Installation library member USERMOD9 provides an example of changing the auto-restore exclusion tables.

Customizing the SAMS:Disk TSS Support

Tape Units Allocated Concurrently

By default, only one tape unit (one restore) is allowed to be allocated by dynamic restore at any given time. Sysparm TSOTULMT may be specified with a numeric value to permit two or more concurrent allocations (restores).

```
TSOTULMT05 tape unit limit set to five
```

Tape Unit Name

The default unit name for allocating a tape drive is “TAPE”. If this unit name does not cover the desired devices, use the following sysparm to provide the correct unit name.

```
TSOTUNITnnnnnnnn where “nnnnnnnn” is the correct name
```

Limiting TSS User Access

The DARCHIVE and DRESTORE commands allow TSS users to archive and restore any data set. To restrict most users to archive and restore only those data sets prefixed with their user IDs, sysparm TSOUSRID must be specified with a value of Y. Access to all data sets may then be given to selected users by placing their user IDs in member TSOUSERI of the parmlib data set.

To allow DARCHIVE requests to have access against offline volumes, sysparm TSOVOLOF must be specified with a value of Y. The default (N) enforces that the data set requested must be found online or the request is rejected. Update parmlib member SYSPARMS with entries TSOUSRIDY and TSOVOLOFY, and create member TSOUSERI with the proper user IDs to accomplish both of these.

TSS Screening Exits for Archive and Restore

Exits are also available that allow an installation to screen archive and restore requests. These exits are documented in the User Exits section in the Systems Guide and may be referenced there if more elaborate restrictions are needed.

It should be noted that both user ID restrictions (sysparm TSOUSRID = Y) and the TSS user screening exits may be used at the same time. The user ID screening check is made first. The user exit module will never see the request if the ID requirement is not met.

Chapter 4. Evaluation

This section presents an introductory approach to evaluating SAMS:Disk, so that you may become familiar with the basic functions, commands and coding conventions. Once these are well in hand, you can explore the full capabilities and multitude of options in each function to come up with an innovative and effective implementation for your specific site. The Examples section in the *User's Guide*, (beginning on page 423) coupled with the functional explanations, should provide you with further ideas and assistance.

As you read these examples, refer to the Data Storage Command Language section, beginning on page 23 in the *User's Guide*. It may aid you in understanding these examples more thoroughly and provide additional ideas in customizing them to better suit your needs.

Testing the SAMS:Disk System

Start by running some sample reports.

Step 1. Run Some SAMS:Disk Reports

For each volume found online, the following JCL will produce summary reports that include such information as volume free space, types of data sets, and data set size distributions. Since these are summary reports, each volume produces only one report line. So, 24 volumes will fit nicely on one page.

```
//jobcard   JOB (acct,info),etc.
//     EXEC DMS
//SYSIN DD  *
SCAN REALVOLS
 VREPORT ALLOCS,ATTRBS,DISTRS
/*
```

Figure 4-1. Sample DSCL Report Commands

If your configuration has a large number of DASD devices, such a scan may require more time than an evaluation may merit. If so, simply limit the report to a selected few volumes by adding the SELECT command to the input stream as follows:

```
//      EXEC DMS
//SYSIN DD *
 SCAN REALVOLS
  SELECT VOL=(PACK01,PACK02)
  VREPORT ALL,ATT,DIS
```

Figure 4-2. Sample DSCL Select Command

Note: Most parameters may be abbreviated to the first three characters. Also, a SAMS:Disk command (for example, SELECT) may start in any column, including column one. If the volume list contains only one entry, no parentheses are needed. Volumes may be specified by the PREFIX followed by a slash (/). For example, to specify all volumes beginning with “PACK..”, use the VOLUME parameter as follows:

```
SELECT VOL=(PACK/)
```

To produce a report of all the data sets on volume “VOS3FS” and any volume that starts with “TSS”, use either of the following:

```
//   EXEC DMS
SCAN REALVOLS
 SELECT VOL=(VOS3FS,TSS/)
  REPORT SVD

// EXEC DMS
 SCAN REALVOLS
  SELECT VOL=(VOS3FS,TSS/)
   REPORT MVD
```

Figure 4-3. Example of All Data Sets on Volume VOS3FS

The SVD report will produce an alphabetic listing of data set names on each volume. That is, each volume’s data sets are listed alphabetically, a new report for each volume. The MVD report will produce an alphabetic listing of data set names on all the volumes. That is, all the data set names are listed alphabetically within a single report.

Step 2. Archive Some Data Sets

The second function of SAMS:Disk evaluated should be explicit archival. Archiving a data set consists of two phases. First, the data is read and stored on a tape or disk that is indexed by SAMS:Disk. Second, disposition for the archived data set is taken. Disposition consists of scratching and uncataloging or recataloging.

The following is an example of archiving a cataloged data set. The data set is scratched and uncataloged as the default disposition.

```
// EXEC DMS
FIND DSN=A.B.C
ARCHIVE
```

In order to modify the disposition action to no-scratch and no-catalog, you must use an ARCHIVE command in the following format:

```
// EXEC DMS
FIND DSN=A.B.C
ARCHIVE DISP=KEEP
```

Note that a full explicit data set name or a data set name pattern may be specified.

Step 3. Restore the Data Sets

Now that you have one or more data sets in the archives, try the RESTORE function with JCL similar to the following:

```
// EXEC RESTORE
RESTORE DSN=A.B.C,VOL=PACK02,NOC
```

If you want the data set restored to a volume other than the one from which it was archived, include the VOL= parameter to direct SAMS:Disk to a particular volume. Default disposition for a restore operation is to catalog the data set following the restore. If the original data set still exists on DASD, you may not want SAMS:Disk to attempt to catalog the data set. Use the NOCATALOG (NOC) parameter to suppress catalog action.

The NEW= parameter will cause SAMS:Disk to restore the original data set under a new name. This would allow you to restore a data set to the same volume from which it was archived without a name conflict. The following example illustrates this:

```
// EXEC RESTORE
RESTORE DSN=A.B.C,NEW=TEMP.B.C
```

Step 4. Submit a Backup Job

The data set backup function can be considered as an automated archival. The operation scans all the data sets on a volume or group of volumes and, based on the criteria you supply, will archive certain data sets and retain others on the volume. This is a very powerful function and, as such, a SIMULATE mode of operation is provided and highly recommended until you are satisfied you will obtain the desired results.

Specification of the retention criteria is very flexible and, as such, is also fairly complex. Review the information in the Data Storage Command Language (beginning on page 23 in the *User's Guide*), BACKUP/ARCHIVE (beginning on page

165 in the *User's Guide*), and Parmlib (beginning on page 437 in the *Systems Guide*) when you set up your own DSCL selection criteria.

```
// EXEC DMS
SET MODE=SIM
SCAN REALVOLS
SELECT VOL=PACK01,
      CRITERIA=(TEMP,EQ,YES,AND,CREDT,LE,TODAY-2)
DELETE
SELECT VOL=PACK01,
      CRITERIA=(MODIFIED,EQ,YES)
BACKUP
```

Figure 4-4. Example of DSCL Scratch and Backup

The first selection criteria will scratch all two-day-old temporary data sets. The second selection criteria will back up any changed VSAM and non-VSAM data sets.

Step 5. Produce a LISTD/LISTV Report

Now you have even more data sets in the archives. If you want to determine what data sets are in the archives, you can run a LISTD, which displays the archive index. The archive index exists as two files, DSNINDEX and ARCHVOLS. The DSNINDEX is keyed on data set name and has one entry for each data set or copy of a data set that resides in the archives. The ARCHVOLS is keyed on volume serial and contains one entry for each archive tape volume. These files may be listed as follows:

```
// EXEC LISTD
LISTD          (DSN=...)
LISTV          (VOL=...)
```

Note that individual data sets or groups may be listed by adding the DSN= parameter to the LISTD, or individual volumes may be listed by adding the VOL= parameter. These may be used as an alternative to listing all of the entries.

Step 6. Run a MERGE Job

Each time an archive or backup run is made, another archive tape is created (or an archive data set on disk, if you have implemented that option). As time passes you will want to consolidate archive tapes (or move the archives on disk off to tape) and get rid of expired data sets.

SAMS:Disk provides an archive MERGE function that will successively process each archive tape volume (or an archive data set on disk) as input. It will copy and distribute only the unexpired data sets according to their expiration dates and the limits you have specified. This allows you to produce archive tape volumes intelligently that will expire some time in the future. That is, you could produce two new archive tape volumes: one that will contain only those data sets that will expire in the coming 30 days, and the other, data sets that will NOT expire in the coming 30 days. This means that 30 days from the run, one tape will be completely expired and will not be mounted for the next merge operation. Also, during the 30-day period, the second tape will contain no expired data.

To perform the forward MERGE of archive data contained on tape volumes, execute a job similar to the following:

```
// EXEC MERGE
//MERGE.SYSIN DD *
MERGE LIMITS=(30,999),PERCENT=70,TYPE=TAPE
```

Note that the MERGE control statement goes in the merge step. Therefore the //MERGE.SYSIN DD * is required with the proper step name.

Step 7. Recover a DASD Volume

After SAMS:Disk has been installed and running for some time, the archives will contain a recent copy of many data sets. If you experience the loss of a DASD volume due to a hardware or software failure, SAMS:Disk can aid you in recovering the lost data. Initialize another DASD volume and perform volume recovery as follows:

```
// EXEC RECOVER
RECOVER VOL=BADPAK,TOVOL=NEWPAK
```

SAMS:Disk will search the archive index and restore any data sets that were archived from the volume BADPAK. If a preallocated data set is found on NEWPAK, SAMS:Disk will treat it based upon the value of sysparm PREALLOC, which you might want to review at this time.

Note that this facility can be used to copy or move data sets, groups of data sets, or entire volumes to another device type. For example, to convert a 8595-1 volume to a 8598 volume, first run backup to ensure that a current copy of all data sets on the 8595-1 volume exists in the archives. Second, run volume recovery to the 8598 volume. SAMS:Disk will recognize the change in device types and recompute the space requirements for each data set.

Step 8. Compress a PDS

SAMS:Disk provides a PDS management facility also. This includes some special PDS reports and a PDS compression utility. The PDS reports are run via the standard SAMS:Disk report procedure and include a PDS Status Report and a PDS Member Report. The member report is a 12-up listing of the member names that reside in the PDS, while the status report includes the data set and directory statistics.

The compression utility can be run explicitly or in a scan mode. The scan mode can evaluate the current status of a PDS to determine if the data set should be compressed. The user specifies certain threshold values that the evaluation module uses to make its determination.

The following step will compress any PDS that resides on volume PACK01 and,

1. is more than 75 percent full

2. has two or more extents

   or

3. has a directory that is more than 90 percent full

```
// EXEC COMPRES
SCAN VOL=PACK01,THRESHOLD=(75,2,90)
```

Step 9. Release Idle Space

If data sets are over-allocated and wasting space, SAMS:Disk provides a flexible and convenient means to free this space for use by others. Thresholds can be established that allow some extra space but, if exceeded, the unused space is released.

The following example releases all of the over-allocated space found in sequential data sets and PDS data sets. (Note that it is not necessary to release all of the over-allocated space, as this example demonstrates.)

```
// EXEC DMS
SCAN REALVOLS
SELECT VOL=PACK01,
       CRITERIA=(DSORG,EQ,PS,OR,DSORG,EQ,PO)
RELEASE
```

Figure 4-5. Example of Over-allocated Space Released

Special Testing Environment

Testing and implementing SAMS:Disk functions that make use of a data set's last used date (for example, archival), or perhaps its last modified date (for example, special backup option), requires some special consideration if the SAMS:Disk SVC has not been installed long enough to have these fields properly updated on all data sets.

The first option is to simply wait an appropriate period of time before implementing a dependent function. The second is to use the DSCB Update Utility (described in Appendix B) to initialize these fields to an appropriate starting value. The third is to use the following sysparms for testing, and then follow either option one or two.

Sysparms DSCBLUSD and DSCBLMOD normally provide the decimal offset into the format-1 DSCB where the last used and last modified dates are being recorded. For testing, however, you might specify them with an offset of either the creation date (decimal offset 53) or the expiration date (decimal offset 56). For your test purposes, these values will cause SAMS:Disk to treat the creation date (or expiration date) as the last used and last modified dates as well. In the example here, the sysparms have been specified with the offset of the creation date:

```
DSCBLUSD053
DSCBLMOD053
```

Note: Be sure to remove these entries once the SAMS:Disk SVC has been active long enough to have recorded proper values. Be sure that you do not have these testing values set if you execute the DSCB Update Utility to initialize the fields!

If a local modification to OPEN is already being used to record either or both of these fields in the DSCB, the sysparm entries can be used to direct SAMS:Disk to the fields maintained by your own local modification. See the descriptions of these system parameters, beginning on page 136 in the *Systems Guide*.

Conclusion of Evaluation

This concludes the beginning evaluation of SAMS:Disk. Many functions and options have not been covered, but we hope that this preliminary testing has served to introduce some techniques whereby SAMS:Disk can be used to deal with data storage management problems. The Examples section in the User’s Guide(beginning on page 423) should provide assistance in constructing specific applications as needed at your installation.

Testing the SAMS:Disk Auto-Restore Feature

There are several good reasons to perform thorough system testing of the auto-restore feature. Some of the benefits that can be derived are:

- You become familiar with the characteristics of the system.
- You will experience what the system does under varying external conditions.
- The system operators will have a chance to get used to the messages that the system issues. They will also have a chance to respond to its operator prompts where they are required.
- If a user screening exit has been written, this provides an excellent opportunity for testing its effectiveness.

- You may formulate some ideas of your own for implementation guidelines.

Forcing an Auto-Restore to be Initiated

To force invocation of the catalog management support, you must first recatalog an archived data set to the SAMS:Disk pseudo-volume. The easiest way to do this is to run an explicit SAMS:Disk archive job with the parameter DISP=RECAT specified. This will cause SAMS:Disk to recatalog the data set after the scratch has taken place.

Once you have the data set recataloged, you can cause the support to be invoked by either running a batch job that references the data set (such as JSDSCPY), or by attempting to edit the data set from a TSS environment. If you do this from an online environment, you will be prompted by several messages:

1. Do you want to restore the data set?,

2. Do you want an immediate or deferred restore?,

   and, if an immediate restore is requested,

3. Do you want to wait for the restore to complete?

Each of these questions was explained more thoroughly under *"Customizing the TSS Auto-Restore Environment"* on page 48.

Testing Auto-Restore in a Production Environment

There are many possible job combinations that can be run to simulate production environments. The following list details possible tests that can be run. Note that the output reports created by the system task DMSAR should be routed to hard copy for your review. All data set references in the examples are to non-existent data sets; that is, no format-1 DSCBs exist for the data set.

1. Run a test specifying a data set that can be found in the archives index, and one that cannot.

2. Execute two auto-restore tasks simultaneously by forcing two jobs to execute at the same time (their job names must be different), each job requesting a different archived data set.

3. Try running the test as in 3 above, but this time reference different data set names that reside on the same archive tape volume (you will have to do a LISTD on an archive volume to find a suitable pair of data sets).

4. If you are supplying an ARESPREX screening exit, or using other options to tailor the function to your environment, try running a few

conditions that test your implementation. Make sure the restore requests are processed as expected.

When you have exercised enough test conditions to feel comfortable with the auto-restore feature, you are ready to move on to planning the implementation. Be sure any changes you made for testing (such as sysparm values or procedure references) are changed back for a production environment.

Appendix A. DSCB Update Utility

This utility is provided to permit you to update most fields in the format-1 DSCB, in both Hitachi and SAMS:Disk defined fields. You can verify your updates very easily by running VTOC reports before and after execution of the update utility. For standard Hitachi fields, either SAMS:Disk reports or Hitachi VTOC listings can be used for the verification. When fields unique to SAMS:Disk are being updated, the changes can be verified by running the DSUTIL report for the respective data sets.

Overview

The format-1 DSCBs to update are selected by data set name or pattern and volume name or pattern. An exclusive enqueue is obtained for each data set unless the DSENQ parameter is supplied. The update utility runs in simulate mode unless the parameter NOSIM is supplied on the input command.

Implicit updates will generate the following message on the operator console:

```
DMS3190 F1DSCB UPDATE UTILITY STARTED, ENTER APPROVAL IF OK
```

To finish executing the update utility, the operator must reply with an “O”. Any other response terminates the update.

Caution

This utility performs the functions of an automated and fairly sophisticated “super-zap” utility for the VTOC. As a result, the benefits and convenience can be very rewarding. However, the potential for causing tremendous damage is equally as great, especially when it is run in implicit mode. Any implicit update should always be run in simulate mode first to ensure you will update only the desired data sets. You should also consider using your security system to restrict access to program ADSDM235, or move that module to a library where even read access is tightly controlled!

Condition Codes

The following condition codes are returned from the DSCB Update Utility:

Table A-1. DSCB Update Utility Condition Codes

Code	Description
0	Updates performed as requested
2	Error in specification of a field value
3	Could not acquire specified DSENQ (that is, EXC or SHR)
4	Security authorization failed for one or more data sets
5	Field name not defined
8	Data set or volume not found
12	Operator denied the request
24	I/O error reading format-1 DSCB

JCL — DSCB Command

The following JCL is required to update one or more DSCBs.

```
//UPDATE  EXEC PGM=ADSMI002,PARM=ADSDM235,REGION=1024K
//STEPLIB  DD  DISP=SHR,DSN=SAMS.DISK.LOADLIB
//ABNLDUMP DD  DUMMY
//CMDPRINT DD  SYSOUT=A
//MSGPRINT DD  SYSOUT=A
//PARMLIB  DD  DISP=SHR,DSN=SAMS.DISK.PARMLIB
//SYSPRINT DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
//SYSIN    DD  *
   DSCB NOSIM,DSNAME=ddd,VOLUME=vvvvvv,......
```

Figure A-1. JCL to Update One or More DSCB's

SYNTAX

One or more DSCB UPDATE commands may be placed following the SYSIN dd statement.

```
DSCB   DSNAME=,VOLUME=,DSENQ=,NOSIM,LUSDT=,LMODT=,JBNM=,CREDT=,
       EXPDT=,OPCNT=,OPCNTH=,LSTVL=,SVCMODE=,PSWRD=,CHBIT=,PTRDS=,
       DSSNO=,VSEQN=,DSORG=,RECFM=,ICFCD=,BLKLN=,LRECL=,KEYLN=,
       RKEYP=,RACFBIT=,L8BLKBIT=,CHKPTBIT=,SCALO=,LSTAR=,TRBAL=,
       ALLOTYP=,NOEPV=,NOBDB=,FIELD=,FROMCOL=,ONLY
```

DSNAME=

The data set name(s) or pattern(s) to supply the key for the format-1 DSCB(s) to be updated. Up to 20 data set or pattern names may be entered.

VOLUME=

This parameter may optionally be specified to designate which volume(s) contain the DSCB when the system catalog is not to be used. If this is an implicit update (pattern matching used), the volume(s) or pattern(s) must be specified. Up to 50 volume or pattern names may be entered.

DSENQ=

The type of enqueue to perform for each data set updated. There are three valid values: EXC for an exclusive enqueue (this is the default), SHR for a shared enqueue and NO for no enqueue.

NOSIM

This parameter is the reverse of the standard SIM parameter. The update utility runs in simulate mode unless this parameter is supplied.

LUSDT=

A date in an accepted SAMS:Disk format to replace the last use date is specified as the value of this parameter. Unless overridden by a value specified in sysparm DSCBLUSD, this date is converted to the discontinuous binary format and is stored at a displacement of 75 bytes.

LMODT=

A date in an accepted SAMS:Disk format to replace the last modified date for non-VSAM data sets is specified as the value of this parameter. Unless overridden by the value specified in sysparm DSCBLMOD, this date is converted to the discontinuous binary format and is stored at a displacement of 70 bytes. VSAM components are ignored for this processing because VSAM modified date information is maintained in the SYSTEM-TIMESTAMP information in the catalog.

JBNM=

The non-VSAM job name or accounting field that referenced the data set is specified as the value of this parameter. Unless overridden by the value specified in sysparm DSCBJBNM, the job name will be stored at a displacement of 62 bytes. VSAM components are ignored for this processing because this field is an Open SVC (non-VSAM only) field.

CREDT=

A date in an accepted SAMS:Disk format to replace the creation date is specified as the value of this parameter.

EXPDT=

A date in an accepted SAMS:Disk format to replace the expiration date is specified as the value of this parameter.

OPCNT=

The cumulative count of opens for the respective non-VSAM data set is specified as the value of this parameter. Unless overridden by the value specified in sysparm DSCBOPCD, it will be stored as a binary fullword at a displacement of 78 bytes. The highest value for this parameter is 99999. VSAM components are ignored for this processing because this field is an Open SVC (non-VSAM only) field.

OPCNTH=

The cumulative count of opens for the respective non-VSAM data set is specified as the value of this parameter. Unless overridden by the value specified in sysparm DSCBOPCD, it will be stored as a binary halfword at a displacement of 78 bytes. VSAM components are ignored for this processing because this field is an Open SVC (non-VSAM only) field.

LSTVL=

Specify ON to cause the data set to be indicated as the last volume on which the data set resides. Specifying OFF will indicate that the data set is part of a multivolume data set, and not the last part. (Sets hex 80 bit of DS1DSIND field at offset dec 93.)

SVCMODE=

Specify the SAMS:Disk SVC MODE with which the DSCB has been updated. This parameter should only be specified during conversion of DSCBs maintained with a SAMS:Disk Open SVC from a release prior to Release 8.1 to DSCBs maintained with a SAMS:Disk Open SVC from a Release 8.1 or higher.

PSWRD=

Specify NO to turn off both the write protect bit and the read/write protect bit. Specify RW to turn on the read/write protect bit. Specify W to turn on the write protect bits (sets the hex 10 and hex 04 bits of DS1DSIND field at offset dec 93).

Note: To change a data set that is currently W-protected to RW-protected, first use a command with NO followed by one with RW.

CHBIT=

Specify ON to turn the change bit on, indicating that the data set contents have changed. OFF turns the change bit off (sets the hex 02 bit of the DS1DSIND field at offset dec 93 as established by SU 60 for VOS3).

PTRDS=

This is the cylinder, track and record in CCHHR form pointing to the format-3 DSCB for the data set. It is a five-byte hexadecimal field and anything from 0000000000 to FFFFFFFFFF is accepted. A value of 0000000000 indicates no format-3 DSCB exists.

DSSNO=

This parameter replaces the data set serial number which is a six-character field.

VSEQN=

This parameter replaces the volume sequence number with a value between 0 and 255.

DSORG=

The data set organization bytes. This parameter is entered as a two-byte hexadecimal value. Anything from 0000 to FFFF is accepted. Standard values are:

8000	—	for indexed sequential (ISAM)
4000	—	for sequential (BSAM, QSAM)
2000	—	for direct (BDAM)
0200	—	for partitioned (BPAM)
0008	—	for VSAM

RECFM=

The record format field. This parameter is a one- byte hexadecimal value. Anything from 00 to FF is accepted. Standard values are:

80	—	bit for fixed
40	—	bit for variable
C0	—	bits for undefined
20	—	bit for track overflow
10	—	bit for blocked
08	—	bit for standard (fixed) or spanned (variable)
04	—	bit for ASA control characters
02	—	bit for machine control characters

ICFCD=

This is the OPTCD field in the DSCB. It indicates whether a data set is an DMF/EF Type-2 catalog or a cluster defined in a DMF/EF Type-2 catalog. It is a one-byte hexadecimal value. Anything from 00 to FF is accepted. Standard values are:

80	—	for a DMF/EF Type-2 catalog
40	—	for cluster defined in a DMF/EF Type-2 catalog

BLKLN=

This is the data set block size. The value range is 0 to 32760.

LRECL=

This is the logical record size for the data set. It can range from 0 to 32760.

KEYLN=

The key length is a one-byte field. The highest value accepted is 255.

RKEYP=

The relative key position is the offset within the data block of the data key. Values from 0 to 32760 are accepted.

RACFBIT=

Specify ON to indicate the data set is RACF- protected, and OFF to remove the RACF protection.

L8BLKBIT=

Specify ON to indicate the block length supplied in the DSCB is given as a multiple of eight bytes and OFF to indicate the block length is the real length in bytes.

CHKPTBIT=

Specify ON to indicate this is a check point data set. Specify OFF to remove this indication.

SCALO=

This parameter replaces the secondary allocation value. The highest value accepted is 99999.

LSTAR=

This is the last relative track and record written for a data set. This parameter is entered as a three-byte hexadecimal value. The first two bytes are the relative track number and the third byte is the record number. Anything from 000000 to FFFFFF is accepted.

TRBAL=

This is the number of bytes remaining on the last track written on a data set. Values from 0 to 32760 are accepted.

ALLOTYP=

The allocation type field. This parameter is a one-byte hexadecimal value. Any value from 00 to FF is accepted. Standard values are:

80	—	bit for track
40	—	bit for block
C0	—	bits for cylinder
00	—	bits for absolute
08	—	bit for contiguous
01	—	bit for rounding

NOEPV=

Number of extents on this volume for the data set. The maximum valid value is 16.

NOBDB=

Number of bytes used on the last directory block for this partitioned data set.

The following parameters allow you to move SAMS:Disk-supported fields from one position in the format-1 DSCB to another. Before using these parameters you should decide on a new offset for a field and update your SAMS:Disk SVC and the appropriate SAMS:Disk sysparm to reflect the new offset. Then supply a list of the SAMS:Disk-defined fields to move and their original offsets in the format-1 DSCB. The target offsets are obtained from the sysparm that corresponds to that SAMS:Disk-defined field. This utility then gets the value from the input offset and stores it at the sysparm-defined offset.

FROMCOL=

A list of offsets for the fields specified in the FIELD parameter. There must be one offset for each field name. Data will be moved from the offsets specified here to the new offsets as specified by the corresponding SAMS:Disk sysparms.

ONLY

This parameter specifies that only DSCBs identified with a SVC MODE value of less than 4 be updated. This parameter is only required on VOS3 systems.

This parameter will prevent the DSCB Update Utility from converting DSCBs that were previously converted.

FIELD=

A list of SAMS:Disk fields to move. Up to seven field names can be supplied. The valid field names are:

Table A-2. List of DSCB Fields that can move

Field	Description
CRJB5	first five bytes of creating job name (pre-R8.1)
CRJB3	last three bytes of creating job name (pre-R8.1)
LUSJB	job name or account field (pre-R8.1)
LMODT	last modified date
LUSDT	last use date
OPCNT	cumulative open count in a fullword
OPCNTH	cumulative open count in a halfword

Each of these fields has a corresponding sysparm that will supply the target offset for the move.

Note: For the open count, if the from count is a fullword value, specify OPCNT as the “from field” name. If it is a halfword value, specify OPCNTH as the “from field” name. If you are changing the open count from a fullword to a halfword, make certain that you specify OPCNT for the fullword from field, and that sysparm DSCBOPCD has an “H” specified in the fourth position (that is, DSCBOPCDxxxH) for the new halfword target field. If you are changing the open count from a halfword to a fullword, be sure that you specify OPCNTH for the halfword from field, and that sysparm DSCBOPCD has a blank specified in the fourth position for the new fullword target field.

User Exits

The user exit DSCBCVEX, described on page 217 in the *Systems Guide*, is also available to aid you when moving SAMS:Disk-support fields.

Appendix B. IXCATLG Utility

This maintenance facility was designed to read the DSNINDEX archive index and catalog those data sets that are no longer on DASD. This permits these data sets to be auto-restored if they are required some time in the future. The facility has several options that allow the user to catalog only a subset of archived data sets, depending on command parameters specified. Simulation mode may be specified to produce a report showing all catalog actions that would take place, without performing any catalog updating.

IXCATLG Function Description

When you run the IXCATLG utility, SAMS:Disk will read the archive index sequentially until all archive records have been processed. Only the most current index record for each data set is examined. If the data set was not scratched when it was archived, the index record is ignored and the data set will not be cataloged.

If parameter VSAMONLY is specified, only data sets marked as VSAM in the index will be processed. Conversely, if NONVSAM is specified, only non-VSAM data sets will be processed. The default is to process all data sets.

If the DSNAME= and/or VOLUME= parameters were specified, the data set name and/or archive volume is matched against the pattern(s) specified. If they both match, or if only one was specified and it matches, the data set is selected for processing. Then a check is made to see if the EXCLUDE parameter was specified. If it was, the selection decision made by the DSNAME= and VOLUME= parameters is reversed. That is, if the data set would have been included it is now excluded, and vice versa.

If the data set has passed all selection testing to this point, a check is made to determine if the DATE= parameter was specified. If it was, the archive date in the DSNINDEX record is compared against the date entered on the command. If the archive date is less than the command date, the data set is bypassed. If the DATE= parameter is not specified, processing continues for the data set.

At this point the data set has gone through all preliminary screening. The user exit IXCTLGEX is now invoked, if specified, with the DSNINDEX record currently being processed and a selection flag. The exit can change the selection flag to either include or exclude the data set. More information regarding the exit can be found below.

If the data set passed all screening requirements, a catalog locate is issued to determine if the data set is currently cataloged. If it is not currently cataloged, an obtain is issued for the format-1 DSCB on the volume from which the data set was archived. If the data set is not on the volume, it is cataloged to the SAMS:Disk pseudo-volume (assuming SIMULATE mode was not specified).

If the data set is currently cataloged, SAMS:Disk checks to see if the cataloged volume is the same as the pseudo-volume. If it is, no processing is required for the data set. If it is not the same volume, SAMS:Disk checks to see if a list of staging volumes was specified (STAGEVOLS= parameter). If none were specified, or if the catalog volume matches one in the STAGEVOLS= list, SAMS:Disk issues an obtain for the format-1 DSCB of the data set on the volume to which the data set is cataloged. If it is not found on that volume, the data set is recataloged to the pseudo-volume (again assuming SIMULATE mode was not specified).

At the end of the run, SAMS:Disk prints some statistics concerning the number of catalog actions required to perform the conversion.

We recommend that this utility be run in a simulate mode initially to verify that the proper data sets will be cataloged. Because of the number of locates, obtains and catalog actions processed by this utility, a significant amount of run time can be expected. The more active your installation is in archival (with scratch specified), the higher the CPU usage will be.

```
//CATLG EXEC IXCATLG
//SYSIN    DD *
 CATALOG SIM,DSN=,VOL=,EXCLUDE,DATE=,EXPIRED,
       FAILSAFE,VSAMONLY,NONVSAM,STAGEVOLS=
```

Figure B-1. IXCATLG JCL

IXCATLG Command and Parameters

SIM — Specify this OPTIONAL parameter to run the utility in simulate mode, which will determine which data sets would be cataloged if this job were run in a live mode. This parameter allows the catalog utility to perform all processing except the final cataloging action. Note that this parameter will not reduce significantly the amount of CPU time required by the maintenance utility.

DSN**AME=** — Use this OPTIONAL parameter to specify from 1 to 50 data set names and/or patterns. If this parameter is specified, only those data sets that match any pattern will be cataloged. This parameter's function can be changed by parameters VOLUME= and EXCLUDE. Refer to these parameters for more details.

VOL**UME=** — Use this OPTIONAL parameter to specify from 1 to 50 volumes and/or patterns. If this parameter is specified, only those data sets that were archived from one of the DASD

	volumes specified will be cataloged. If the DSNAME= parameter is also coded, the data set must match one of the data set name patterns specified and it must have been archived from one of the volumes specified. This parameter's function can be changed by parameter EXCLUDE.
EXC**LUDE**	Use this OPTIONAL parameter in conjunction with the DSNAME= and/or the VOLUME= parameters. When this parameter is specified, the patterns entered for DSNAME= and VOLUME= become exclusion lists and only those data sets NOT matching the patterns will be cataloged. EXCLUDE has no effect on any parameters other than those mentioned.
DATE=	Specify this OPTIONAL parameter to limit the data sets being cataloged to those that were archived on or after the date specified. The date may be specified in any standard SAMS:Disk format desired.
EXPIRED	Specify this OPTIONAL parameter to allow cataloging of expired data sets. Default processing excludes expired data sets from being cataloged.
FAILSAFE	This is a REQUIRED parameter. It serves no function in the recataloging facility, but was added as a failsafe mechanism to prevent users with "old" JCL from submitting the same job stream without performing updates. If this parameter is not supplied, the job will fail with a command input error.
NONVSAM	You may specify this OPTIONAL parameter to force SAMS:Disk to process only non-VSAM data sets. Default processing is that all data sets are eligible.
VSAMONLY	You may specify this OPTIONAL parameter to force SAMS:Disk to process only VSAM data sets. Default processing is that all data sets are eligible.
STAGEVOLS=	You may specify up to ten volume names. This OPTIONAL parameter was designed for SAMS:Disk users who used the auto-restore capability in a prior release (pre-7.1) and recataloged archived data sets to staging volume(s). This parameter is intended as a means of reducing the CPU requirements of this job. When this parameter is specified, an obtain for the format-1 DSCB is attempted only if the data set is currently cataloged to one of the volumes specified in the STAGEVOLS= parameter.

IXCTLGEX - IXCATLG User Screening Exit

This screening exit allows the user to modify individually the data set selection made by the IXCATLG maintenance utility. This exit is invoked after all selection tests have been made, based on the input parameters specified by the user.

It is passed two parameters:

1. the DSNINDEX record being processed

2. the selection flag field

The selection flag is set to either a Y or N when the exit is invoked, Y meaning the data set will be cataloged, N meaning it will be bypassed. Based on any information in the DSNINDEX record, the screening exit may change it to either a Y or N value.

Sample source for the IXCTLGEX module is supplied in the SAMS:Disk installation library. It does no processing by default. If you wish to code your own exit, you may modify this version of the module. For additional information, review the IXCTLGEX User Exit description on page 225 of the *Systems Guide*.

Appendix C. Files Data Set Conversion

If your current release of SAMS:Disk is 8.0.D or below, you must convert your files data set into a different format. The following steps describe how to accomplish the conversion.

1. Make a backup copy of the current files data set.

2. Run an UNLOAD pointing to the current files data set and the current LOADLIB. Review the following procedure, located in the SAMS:Disk 9.1 PROCLIB, before execution.

```
//UNLOAD   PROC MI=000,
//              Q='SAMS.DISK.RNN',
//              S='*',
//              W=SYSDA
//UNLOAD   EXEC PGM=ADSMI&MI,PARM='ADSDM177',REGION=4096K
//*******************************************************
//*
//*          SAMS:DISK FILES UNLOAD - DEFAULT IS TO DISK
//*
//*******************************************************
//STEPLIB  DD  DISP=SHR,DSNAME=&Q..LOADLIB
//ABNLDUMP DD  SYSOUT=&S
//CMDPRINT DD  SYSOUT=&S
//FILES    DD  DISP=SHR,DSNAME=&Q..FILES
//MSGPRINT DD  SYSOUT=&S
//PARMLIB  DD  DISP=SHR,DSNAME=&Q..PARMLIB
//SYSPRINT DD  SYSOUT=&S
//SYSUDUMP DD  SYSOUT=&S
//SEQFILES DD  DISP=(NEW,CATLG,DELETE),DSN=&Q..SEQFILES.UNLOAD,
//             DCB=(LRECL=268,BLKSIZE=6144,DSORG=PS,RECFM=VB),
//             SPACE=(6144,(360,360),RLSE),UNIT=&W
```

Figure C-1. UNLOAD procedure

You may also use the following JCL.

```
//UNLOAD   EXEC UNLOAD,Q='SAMS.DISK.Rnn'
//FILES    DD  DISP=SHR,DSN=current.files.dsn
//SYSIN    DD  *
UNLOAD ALL
```

Figure C-2. UNLOAD JCL

3. Estimate the capacity needs of the files data set. Review the *"Estimating Subfile Capacity Requirements"* on page 11 of the *Systems Guide*.

- The physical space your files data set resides on will increase.
- This increase occurred when the logical record length of the DSNINDEX subfile changed in release 8.1 from 090 to 256.
- Calculate your new space requirements for the files data set. Review the *"Estimating Physical Space Requirements"* on page 11 of the *Systems Guide*.

4. Create a FILEDEFN member in your Release 9.1 parmlib.

   - You can use a FILEDEFN member from a prior release if you insure the LRECL and KEY lengths for the following entries are adjusted as shown:

     ```
     DSNINDEX256044C00075000YNNY
     RESTCMDS164044C00005000YYNN
     ```

   - You can create a new FILEDEFN member. If you choose to create a new FILEDEFN member, use member FDSAMPLE in the Release 9.1 parmlib as a sample.

5. Run a FILEINIT, using the FILEDEFN member you choose is step 4, to allocate and initialize the files data set. The following sample JCL can also be found as member FILEINIT in the installation library.

```
//FILEINIT EXEC PGM=ADSMI002,PARM='ADSDM100'
//STEPLIB  DD  DISP=SHR,DSN=SAMS.DISK.LOADLIB
//ABNLDUMP DD  DUMMY
//CMDPRINT DD  SYSOUT=A
//FILES    DD  DSN=SAMS.DISK.FILES
//             DISP=(,CATLG,DELETE),
//             UNIT=SYSDA,
//             DCB=(DSORG=DA),
//             SPACE=(CYL,10,,CONTIG)
//MSGPRINT DD  SYSOUT=A
//PARMLIB  DD  DISP=SHR,DSN=SAMS.DISK.PARMLIB
//SYSPRINT DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
```

Figure C-3. FILEINIT JCL

6. Run a RELOAD pointing to the UNLOADed files data set (created in step 2) and the 9.1 formatted files data set (created in step 5) using the 9.1 LOADLIB.

```
//RELOAD   PROC MI=002,
//              Q='SAMS.DISK.RNN',
//              S='*',
//RELOAD   EXEC PGM=ADSMI&MI,PARM='ADSDM192',
//              REGION=4096K
//* *****************************************************
//**                                                    *
//**           SAMS:DISK FILES RELOAD                   *
//**                                                    *
//* *****************************************************
//STEPLIB  DD  DISP=SHR,DSNAME=&Q..LOADLIB
//ABNLDUMP DD  SYSOUT=&S
//CMDPRINT DD  SYSOUT=&S
//FILES    DD  DISP=SHR,DSNAME=&Q..FILES
//MSGPRINT DD  SYSOUT=&S
//PARMLIB  DD  DISP=SHR,DSNAME=&Q..PARMLIB
//SYSPRINT DD  SYSOUT=&S
//SYSUDUMP DD  SYSOUT=&S
//SEQFILES DD  DISP=OLD,DSN=&Q..SEQFILES.UNLOAD
```

Figure C-4. RELOAD procedure

You may also use the following JCL.

```
//RELOAD     EXEC  RELOAD
//SYSIN      DD  *
RELOAD ALL,FORMAT
```

Congratulations! Your files data set has now been converted.

Appendix D. S213 Abend Exit

"F1-DSCB Not Found" S213 Abend Exit (IFG0EX0A Exit)

This exit is not highly recommended because some installations, particularly those using SMS, are less likely to have hard coded volsers in JCL. Also, using this exit can result in restore errors because the time at which it is called does not allow for data set allocations to change volumes. These changes are more likely to occur in an SMS environment.

The recommendation to not use this exit is twofold:

1. The methods that would cause the exit to be invoked are not generally practical and they are not compatible within an SMS environment. These methods are:

    a. Archiving data sets and not changing the volume indicator in the catalog to reflect that they are archived.

    b. Coding the VOLSER in JCL to allocate to data sets that are already in existence.

2. The S213 exit has the following significant limitations:

    a. Only non-VSAM data sets can be auto-restored.

    b. No pooling capabilities.

    c. Restores can not be invoked by ISPF.

    d. Installation requires an IPL.

    e. False invocation of this exit has been know to occur.

The Catalog Management hook (discussed on page 34 in this manual) has the following advantages over this exit:

1. VSAM and non-VSAM data sets can be auto-restored.

2. Restored data sets can be pooled by SAMS:Disk, SAMS:Allocate or DFSMS.

3. Restores can be invoked by ISPF.

4. Installation and/or removal of the Catalog Management hook is accomplished dynamically with a started task.

The first exit that SAMS:Disk supported was the “F1-DSCB-not-found” user exit in OPEN and EOV processing. The exit is also called the “S213 abend” exit. It is invoked during OPEN processing (non-VSAM) when the data set being opened is cataloged to a volume, but no F1-DSCB exists on that volume for that data set. It is also invoked by batch jobs when the volume of a data set is hard-coded in the JCL, but does not exist on the volume.

At this point the exit is invoked as a last-ditch effort to save the job from a system 213 abend. If the data set is restored to the volume by the exit, it can signal OPEN to retry the DSCB search again; since it just restored the data set, there should be one out there! If it couldn’t restore the data set, the job gets the S213 abend. The dummy exit that IBM supplies with the base MVS system merely sets the return code to tell OPEN to go ahead and abend. The exit name that gets called is IFG0EX0A.

So why the need for two different methods? Because neither of them alone can initiate auto-restores at all times when they are needed. The IFG0EX0A exit does not get called for VSAM data sets, nor does it get invoked for TSO/ISPF functions--the catalog management hook does, however. So why not implement just the catalog management support? If you only did that, you could not automatically restore data sets in batch jobs that had the volume hard-coded in the JCL, or when OPEN is processing a data set that is cataloged to a real volume on which the data set doesn’t reside (F1-DSCB missing!). Conceptually you can think of the two methods as complementary functions. Whenever processing for a data set is going to occur, two primary questions must be answered:

1. What volume does the data set reside on?

2. What are its attributes on the volume?

To answer the first question, catalog management is invoked with a locate option, which asks where the data set is currently located. It will search its catalogs until it finds the entry for the data set. With this information, OPEN can allocate the volume and then perform an obtain for the F1-DSCB to get the data set’s attributes and extent information. By understanding the order of information retrieval you will better understand how these methods work, and also why the catalog management hook allows more flexibility.

Assume in the above example that the data set is cataloged to a real volume instead of the SAMS:Disk pseudo-volume. If the data set was being opened for processing, a catalog locate would be issued to determine the location of the data set. Since the data set is cataloged to a real volume, the catalog management hook would let the locate pass through without any modifications. OPEN processing would then allocate the volume(s) for the data set. It would then issue an obtain for the F1-DSCB that describes that data set. If this obtain fails, the IFG0EX0A exit gets control to determine if it can restore the data set. If it can restore the data set at this point, however, it must go back to the volume to which it is cataloged. Remember, OPEN has already allocated the volume and is expecting the data set on that volume only. This is why the catalog management hook allows more flexibility--it intercepts the locate before OPEN allocates the volume(s). It can therefore put the data set back to any volume it desires, as long as it passes back the correct volume list to the requestor of the locate.

Installing the S213 Abend Exit

Installing the S213 exit requires an IPL.

- If you installed module IFG0EX0A into your LPALIB from a release of SAMS:Disk prior to Release 7.1, you will need to remove it as follows.
  - Assemble and link member IFG0EX0A, contained in the SAMS:Disk installation library, into your SYS1.LPALIB.
  - If you also have DFHSM and want SAMS:Disk and DFHSM to auto-restore data sets, use the IFG0EX0A supplied by DFHSM.
- If you did not install module IFG0EX0A into your LPALIB from a release of SAMS:Disk prior to Release 7.1, continue here.
- Sample JCL is provided in the installation library member USERMOD7.
- After completing the relink, run the following JCL to verify your changes.

```
//LIST    EXEC PGM=AMBLIST
//SYSPRINT  DD SYSOUT=*
//SYSLIB    DD DISP=SHR,DSN=SYS1.LPALIB
//SYSIN     DD *
  LISTLOAD MEMBER=(IFG0EX0A)
```

Figure D-1. JCL to Verify Relink Changes

Make sure that the correct version of IFG0EX0A has been link-edited. If DFHSM is not installed, the IFG0EX0A CSECT in module IFG0EX0A should consist of only two instructions:

  - SR 15,15
  - BR 14.

- IPL your system with either the CLPA or MLPA option. The auto-restore function is now active. If MLPA is used, be sure to add the proper entry to SYS1.PARMLIB member IEALPAxx and specify it during the IPL.

Note (an easy mistake to make): When changing modules that reside in LPA, it may be common practice to link the new version into a linklist library other than SYS1.LPALIB, and then use the MLPA ability to test the new module. If all testing goes well, you might use IEBCOPY to copy and replace the old version in SYS1.LPALIB, and then schedule an IPL with a CLPA. This will all work fine, but only if you specify all of the alias names as well as the true member name in the SELECT MEMBER=(...list...) statement for IEBCOPY. Omitting the alias names can cause the IPL with CLPA to fail with unpredictable symptoms. This can be very confusing, because “The MLPA worked just fine! How can the CLPA possibly not work?”

If archived data sets are cataloged to the SAMS:Disk pseudo-volume, they can be restored by the catalog management hook to any one of a pool of volumes. If the data set is recataloged to a real volume, only the S213 exit of OPEN will be invoked and the restored data set will have to go back to that specific volume. The real volume to which archived data sets are recataloged may be thought of as a common “staging” volume for auto-restores. The volume can be monitored for activity and easily managed to ensure enough space to handle the restores.

If the S213 exit is installed, the following restrictions will apply:

1. The IFG0EX0A exit does not get invoked for VSAM data sets, and therefore they must be restored manually.

2. The data set must be restored to the same volume to which the 213 was invoked. If sufficient free space does not exist on the volume, the restore fails and the originating task will terminate with a system 213 abend.

3. The IFG0EX0A exit is not entered from TSO and ISPF functions.

4. The IFG0EX0A exit is not entered from the IEBCOPY utility.

Index